Ollscoil na Gaillimhe
University of Galway

Provided by the author(s) and University of Galway in accordance with publisher policies. Please cite the published version when available.

Title	Next generation HBBTV services and applications through multimedia synchronisation
Author(s)	Yuste, Lourdes Beloqui
Publication Date	2015-09-18
Item record	http://hdl.handle.net/10379/5265

Downloaded 2024-04-23T15:15:00Z

Some rights reserved. For more information, please see the item record link above.

Next Generation HBBTV Services and Applications Through Multimedia Synchronisation

Lourdes Beloqui Yuste

Discipline Information Technology

National University of Ireland, Galway

A thesis submitted for the degree of

PhD

Supervisor: Dr. Hugh Melvin
Dean of Engineering and Informatics: Prof. Gerry Lyons
External Examiner: Dr. Christian Timmerer

Contents

Contents i

List of Figures vi

List of Tables x

Nomenclature xxi

Abstract xxiii

Papers Published xxiv

0.1 Pending Submission/Acceptance . . . xxiv
0.2 Accepted . . . xxiv
0.3 Other Publications . . . xxv

1 Introduction 1

1.1 IP Network Media Delivery Platform . . . 1
1.1.1 IPTV . . . 1
1.1.2 Internet TV/Radio . . . 2
1.1.3 HbbTV . . . 2
1.2 Multimedia Synchronisation Research . . . 3
1.3 Research Motivation . . . 3
1.4 Research Questions . . . 4
1.5 Solution approach . . . 5
1.6 Thesis Scope . . . 5
1.7 Contribution of this thesis . . . 6
1.8 Thesis outline . . . 6

2 Media Delivery Platform, Media Containers and Transport Protocols 8

2.1 QoS/QoE . . . 9
2.2 IP Network Platform . . . 10
2.2.1 IPTV . . . 10

2.2.1.1 IPTV Media Content . . . 11
2.2.1.2 IPTV Functions and Services . . . 12
2.2.1.3 IPTV Main Structure . . . 12
2.2.1.4 IPTV Communications Protocols . . . 15
2.2.2 Internet TV . . . 17
2.2.2.1 Codecs for Internet TV . . . 18
2.2.2.2 Media Delivery Protocols . . . 18
2.2.3 HbbTV . . . 19
2.2.3.1 HbbTV Functional Components . . . 20
2.2.3.2 Formats . . . 21
2.2.3.3 Protocols . . . 23
2.2.3.4 Applications . . . 23
2.2.3.5 HbbTV video/audio . . . 24
2.2.3.6 RTSP . . . 24
2.2.3.7 SDP . . . 26
2.3 Media Containers . . . 27
2.3.1 MPEG-2 part 1: Systems . . . 27
2.3.2 MPEG-4 part 1: Systems . . . 31
2.3.2.1 Architecture . . . 31
2.3.2.2 Terminal Model . . . 31
2.3.2.3 Object Description Framework . . . 33
2.3.2.4 T-STD . . . 38
2.3.3 MPEG-4 part 12: ISO Base Media File Format . . . 42
2.3.4 MP3 Audio File Format . . . 44
2.3.5 DVB-SI and MPEG-2 PSI . . . 48
2.3.5.1 DVB-SI . . . 48
2.3.5.2 MPEG-2 PSI . . . 48
2.3.5.3 DVB-SI Time related Tables . . . 51
2.3.6 MMT . . . 52
2.4 Transport Protocols . . . 55
2.4.1 RTP (Real-Time Transport Protocol) . . . 55
2.4.1.1 RTP Timestamps . . . 56
2.4.2 RTCP (Real-Time Control Protocol) . . . 56
2.4.2.1 RTCP Packets Fields Related to QoS . . . 59
2.4.2.2 Analysing Sender and Receiver Reports . . . 60
2.4.3 RTP Payload for MPEG Standards . . . 60
2.4.3.1 RFC 2250: RTP Payload for MPEG-1/MPEG-2 . . . 60
2.4.4 RTP issues with Internet Media Delivery . . . 63
2.4.4.1 Issues relating RTP over UDP with NAT/Firewalls . . . 64

2.4.5 MMT versus RTP and MP2T . . . 65
2.4.6 HTTP Adaptive Streaming . . . 66
2.4.6.1 HTTP Adaptive Streaming . . . 66
2.4.6.2 MPEG-DASH . . . 67
2.5 Summary . . . 70
2.5.1 Media Delivery Platforms . . . 70
2.5.2 Media Containers . . . 71
2.5.3 Transport Protocols . . . 71

3 Multimedia Synchronisation 72
3.1 Clocks . . . 72
3.1.1 Delivering Clock Sync (NTP/GPS/PTP) . . . 73
3.1.2 Clock signalling . . . 74
3.2 Media synchronisation . . . 76
3.2.1 Multimedia Sync Types . . . 77
3.2.2 Intra-media Synchronisation . . . 78
3.2.3 Inter-media Synchronisation . . . 78
3.2.3.1 Types Inter-media Synchronisation . . . 78
3.3 Synchronisation methods . . . 80
3.4 Synchronisation Threshold . . . 80
3.5 Sampling Frequency . . . 82
3.6 MP2T Timelines . . . 83
3.6.1 T-STD . . . 84
3.6.2 Clock References . . . 86
3.6.2.1 Clock References within MP2T Streams . . . 86
3.6.2.2 Encoder and decoder sync . . . 89
3.6.3 Timestamps . . . 92
3.6.3.1 Timestamp Errors . . . 96
3.6.4 ETSI TS 102 034: Transport MP2T Based DVB Services over IP Based Networks. MPEG-2 Timing Reconstruction . . . 96
3.7 MPEG-4 Timelines . . . 98
3.7.1 STD . . . 98
3.7.2 Clock References . . . 99
3.7.2.1 Mapping Timestamps to the STB . . . 102
3.7.2.2 Clock Reference Stream . . . 103
3.7.3 Timestamps . . . 103
3.8 ISO Timelines . . . 105
3.8.1 ISO Time Information . . . 105
3.8.2 Timestamps within ISO . . . 108
3.9 MPEG-DASH Timelines . . . 110

3.10 MMT Timelines . . . 113
3.11 Multimedia Sync. Solutions and applications . . . 114
3.11.1 Media Delivery . . . 114
3.11.2 Applications . . . 116
3.11.3 Inter-destination Media Sync via RTP Control Protocol . . . 118
3.11.4 Multimedia Sync. HBB-NEXT Solution (Hybrid Sync) . . . 121
3.11.4.1 TVA_id Descriptor . . . 123
3.11.4.2 Broadcast Timeline Descriptor . . . 123
3.11.4.3 Time Base Mapping Descriptor . . . 124
3.11.4.4 Content Labelling Descriptor . . . 124
3.11.4.5 Synchronised Event Descriptor . . . 125
3.11.4.6 Synchronised Event Cancel Descriptor . . . 125
3.12 Summary . . . 125

4 Prototype Design **128**
4.1 Research Questions . . . 128
4.2 High Level Solution Architecture . . . 129
4.2.1 From High Level to Prototype . . . 129
4.3 Detailed Prototype Description . . . 130
4.3.0.1 Server-side Threads . . . 131
4.3.0.2 Client-side Threads . . . 134
4.4 Technology used . . . 134
4.5 Media files used . . . 135
4.5.1 Event . . . 135
4.5.2 Video . . . 135
4.5.3 Audio . . . 135
4.6 Solution Design . . . 136
4.6.1 Audio Channel Substitution . . . 136
4.6.2 Audio Channel Addition . . . 137
4.7 Media Delivery Protocols . . . 138
4.7.1 IPTV Video Streaming . . . 138
4.7.2 Internet Radio Audio Streaming . . . 138
4.8 Bootstrapping. Sport Event Initial Information . . . 138
4.9 Initial Sync . . . 139
4.9.1 MP2T Work-flow . . . 140
4.9.2 MP3 Work-flow . . . 142
4.10 MP2T Clock Skew Detection . . . 145
4.11 MP3 Clock Skew Detection and Correction . . . 146
4.11.1 MP3 Clock Skew Detection . . . 148
4.11.1.1 Clock Skew Detection by Means of MP3 Frame Size . . . 149

4.11.1.2 Method 1: Clock Skew detection by means of Sampling Bit Rate via RTP with latter derived from wall-clock time . . . 150
4.11.1.3 Method 2: Clock Skew detection by means of RTCP . . . 150
4.11.2 MP3 Clock Correction . . . 150
4.11.2.1 Thresholds for MP3 Clock Skew Correction . . . 152
4.11.2.2 Correction Every Second by a Variable Number of Bytes . . . 153
4.11.2.3 Correction by an MP3 Frame in Variable Time Period . . . 156
4.12 Video and Audio Multiplexing (into a single MP2T Stream) and Demultiplexing 157
4.13 Summary . . . 160

5 Prototype Testing 162
5.1 Testing Overview . . . 162
5.2 Testing . . . 163
5.2.1 Initial Synchronisation . . . 163
5.2.2 Testing MP2T Clock Skew Detection . . . 164
5.2.2.1 MP2T clock skew addition to media file at server-side . . . 167
5.2.3 Testing MP3 Clock Skew Detection and Correction . . . 167
5.2.4 Multiplexing into a final MP2T stream . . . 173
5.3 Prototype as proof-of-concept on single device . . . 174
5.4 Patent Search . . . 174
5.5 Summary . . . 175

6 Contributions, Limitations, and Future Work 176
6.1 Introduction . . . 176
6.2 Core Contributions . . . 176
6.3 Limitations and Future Work . . . 178
6.4 Summary . . . 180

Appendix A. IPTV Services, Functions and Protocols 181

Appendix B. DVB-SI and MPEG-2 PSI Tables 186

Appendix C. Clock References and Timestamps in MPEG 192

Appendix D. DVB-SI and MPEG-2 PSI tables in used prototype 198

Appendix E. RTP Timestamps used in prototype for MP3 streaming 204

Appendix F. ETSI 102 823 Hybrid Sync solution tables 209

Appendix G. Multi bitrate analysis MP2T media files 215

References 217

List of Figures

2.1 Media Content value chain in OIPF [4] . . . 11
2.2 Functional architecture for IPTV Services in OIPF [5] . . . 13
2.3 DVB-IPTV protocols stack based on ETSI TS 102 034 [8] . . . 15
2.4 HbbTV High Level architecture. Figure 2 in [22] . . . 21
2.5 Media Delivery Protocols Stack with RTP, MPEG-DASH and MMT. Green: RTP and HTTP; grey for MP2T/MMT packet and blue PES and MPU packets . 24
2.6 RTSP communications with RTP/RTCP media delivery example . . . 25
2.7 RTSP Format Play Time [27] . . . 25
2.8 RTSP Absolute Time [27] . . . 26
2.9 SDP Main Syntax Structure . . . 26
2.10 Process to packetised a PES into MP2T packets. Multiple MP2T packets are needed to convey one PES . . . 29
2.11 MP2T Header and fields . . . 30
2.12 MPEG-4 Terminal Architecture. Figure 1 in [33] . . . 32
2.13 Object and Scene Descriptors mapping to media streams. Figure 5 in [33] . . . 34
2.14 Example BIFS (Object and Scene Descriptors mapping to media streams) following example Figure 2 from http://mpeg.chiariglione.org/ . . . 35
2.15 Main Object Descriptor and related ES Descriptors . . . 36
2.16 Block Diagram of VO encoders following the example in 2.14 based on Figure 2.14 in [34] . . . 37
2.17 Transport System Target Decoder (T-STD) for delivery of ISO/IEC 14496 program elements encapsulated in MP2T. Figure 1 in [30]. The variables in T-STD are described in Table 2.13 . . . 39
2.18 ISO File Structure example . . . 43
2.19 ISO File system used by MS-SSTR [35] . . . 44
2.20 ISO File example structure and box content . . . 45
2.21 MP3 Header structure . . . 46
2.22 DVB-SI and MPEG-2 PSI relationship tables [40] . . . 49
2.23 DVB-SI and MPEG-2 PSI distribution in a MP2T stream . . . 49

2.24 MMT Architecture from [44] . . . 53
2.25 Relationship between MPU, MFU and media AUs . . . 53
2.26 MMT Logical Structure of a MMT Package [45] . . . 54
2.27 MMT Packetisation [45] . . . 54
2.28 Comparison of Transmitting Mechanisms of MMT in Broadcasting Systems based on Table II from [46] . . . 55
2.29 Relationship of an MMT package' storage and packetised delivery formats [43] . . . 55
2.30 RTP Media packet [47] . . . 56
2.31 RTCP Sender Report packet [47] . . . 58
2.32 RTCP Receiver Report packet [47] . . . 59
2.33 MP2T conveyed within RTP packets and the mapping between RTP timestamp with the RTCP SR NTP wall-clock time . . . 61
2.34 High Level RFC 2250 payload options for ES payload . . . 62
2.35 Example of connection media session highlighting NAT problems [50] . . . 65
2.36 MMT protocol stack [46] . . . 66
2.37 MPD file example . . . 69
2.38 MPEG-DASH Client example from [59] . . . 70

3.1 Intra and Inter-media sync related to AUs from two different media streams. MediaStream_1 contains AUs different length and MediaStream_2 has AUs constant length . . . 79
3.2 Lip-Sync parameters [79] . . . 81
3.3 Video Synchronisation at decoder by using buffer fullness. Figure 4.1 in [34] . . . 83
3.4 Video Synchronisation at decoder through Timestamping. Figure 4.2 in [34] . . . 84
3.5 Constant Delay Timing Model. Figure 6.5 in [84] . . . 84
3.6 Modified diagram from Figure 5.1 in [34]. A diagram on video decoding by using DTS and PTS . . . 85
3.7 Transport Stream System Target Decoder. Figure 2-1 in [30]. Notation is found Table 3.9 . . . 86
3.8 MP2T and PES packet structure . . . 88
3.9 A model for the PLL in Laplace-transport domain modified. Figure 4.5 in [34] . . . 90
3.10 Actual PCR and PCR function used in analysis. Figure 2 in [85] . . . 91
3.11 A GOP high level distribution . . . 93
3.12 A GOP High Level distribution with MP2T timestamps (DTS and PTS) and clock references (PCR) . . . 93
3.13 Association of PCRs and RTP packets. Fig A.1 in ETSI 102 034 [8] . . . 97
3.14 System Decoder's Model for MPEG-4. Figure 2 in [33] . . . 99
3.15 MPEG-4 SL Descriptor. Time Related fields . . . 100
3.16 MPEG-4 Clock References location . . . 101

3.17 VO in MPEG-4 and the relationship with timestamps (DTS and CTS) and clock references (OCR) . . . 101
3.18 M4Mux Descriptor . . . 102
3.19 ISO File System example with audio and video track with time related fields . . . 105
3.20 ISO File System for timestamps related boxes [12] . . . 109
3.21 MPD example with time fields from [89] . . . 111
3.22 MPD example with time fields using Segment Base Structure from [89] . . . 112
3.23 MPD example with time fields using Segment Template from [89] . . . 112
3.24 MPD examples with time fields using Segment Timeline from [89] . . . 113
3.25 MMT Timing system proposed in [91] . . . 114
3.26 MMT model diagram at MMT sender and receiver side [91] . . . 114
3.27 IDMS Architecture Diagram from [102] . . . 118
3.28 Example of a IDMS session. Figure 1 in [102] . . . 119
3.29 RTCP XR Block for IDMS [102] . . . 120
3.30 RTCP Packet Type for IDMS (IDMS Settings) [102] . . . 120
3.31 High Level broadcast timeline descriptor insertion [110] [111] . . . 122
3.32 High Level DVB structure of the HbbTV Sync solution . . . 122
3.33 Links between timeline descriptors fields to implement the direct, from Fig. D.1 in [106], and offset, from Fig. D.2 in [106], broadcast timeline descriptors . . . 124
3.34 Example *content_labelling_descriptor* using *broadcast_timeline_descriptor*. Fig. D.3 in [106] . . . 125
3.35 *Content_labelling_descriptor* using *time_base_mapping* and *broadcast_timeline_descriptor* example. Fig. D.4 in [106] . . . 126

4.1 High Level Diagram of System Architecture . . . 129
4.2 Prototype illustrated within HbbTV Functional Components. Figure 2 in [22] with added proposed MediaSync module . . . 131
4.3 High Level Java prototype. Threads, client and media player . . . 132
4.4 High Level description of the MediaSync Module . . . 132
4.5 High Level diagram showing relationship between RTP and PCR in [8] . . . 133
4.6 High Level DVB table structure of the prototype. In blue the video and two audio streams definitions . . . 137
4.7 Initial Sync performed in the MP2T video stream at client-side. Terms found in Table 4.3 . . . 140
4.8 Initial Sync performed in the MP2T video stream at client-side. Terms found in Table 4.3 . . . 141
4.9 Initial Sync performed in the MP3 video stream at client-side. Terms found in Table 4.4 . . . 143
4.10 Initial Sync performed in the MP3 audio stream at client-side.Terms found in Table 4.4 . . . 144

4.11 MP2T Encoder's and RTP packetiser clocks . . . 145
4.12 Flowchart MP2T Clock Skew detection mechanism . . . 147
4.13 MP3 Encoder's and RTP packetiser clocks . . . 148
4.14 Common MP3 Clock Skew Correction Technique for the two MP3 Clock Skew detection techniques applied . . . 148
4.15 MP3 Clock Skew Detection Work-flow . . . 151
4.16 MP3 Flow Chart Clock Skew Set Level . . . 152
4.17 MP3 Correction thresholds applied in prototype . . . 153
4.18 MP3 8 bits clock skew correction distributed within the MP3 Frame. The bits in green show the MP3 Frame Header. Bits coloured in red show the bits added/deleted within the frame . . . 155
4.19 MP3 entire byte correction within a MP3 Frame. The bits in green show the MP3 Frame Header the byte in red is the byte to added/deleted in the clock skew correction model . . . 156
4.20 MP3 Clock Skew Correction based on a fixed MP3 frame . . . 156
4.21 MediaSync work-flow for audio substitution replacing original audio with the new audio stream . . . 157
4.22 MediaSync work-flow for audio addition adding the new audio stream keeping the original one . . . 158
4.23 Audio packets distribution in the MP2T stream. Original audio (PID=257) and new added audio (PID=258) . . . 159
4.24 High Level demultiplexing structure of DVB-SI and MPEG-2 PSI tables. Following Figure 1.10 in [34] . . . 160

5.1 Visualisation of result from Table 5.2 . . . 166
5.2 Visualisation of the MP3 clock detection and correction results from Table 5.4 . 172

1 RTP RET Architecture and messaging for CoD/MBwTM services overview. Figure F.1 in [8] . . . 182
2 RTP RET Architecture and messaging for LMB services: unicast retransmission. Figure F.2 in [8] . . . 182
3 RTP RET Architecture and messaging for LMB services: MC retransmission and MC NACK suppression. Figure F.3 in [8] . . . 183

4 MP2T packetisation scheme PCR-unaware within AAL5 PDUs [117] . . . 193
5 MP2T packetisation scheme PCR-aware within AAL5 PDUs [117] . . . 193
6 Two PCR packing schemes for AAL5 in ATM Networks. Figure 4.8 in [34] . . . 193

List of Tables

2.1 Differences between IPTV and Internet TV . . . 11
2.2 Video and Audio Codecs within MPEG Standards . . . 18
2.3 Sample of Media Containers used in Internet . . . 19
2.4 Application Information Section. Taken from Table 16 in [24] . . . 22
2.5 Systems Layer formats for content services. Table 6 in [25] . . . 23
2.6 SDP parameters . . . 27
2.7 MPEG-2 Program Stream Structure. Table 2-31 in [30] . . . 28
2.8 MPEG-2 Pack Structure. Table 2-32 in [30] . . . 28
2.9 Pack Header Structure. Table 2-33 in [30] . . . 29
2.10 MPEG-2 Transport Stream Structure. Table 2-1 in [30] . . . 30
2.11 MPEG-2 Transport Stream Packet Structure. Table 2-2 in [30] . . . 30
2.12 DecoderConfig Descriptor [33] . . . 38
2.13 Notation of variables in the MPEG-4 T-STD [30] for Fig. 2.17 . . . 40
2.14 ISO/IEC defined options for carriage of an ISO/IEC 14496 scene and associated streams in ITU-T Rec. H.222.0. ISO/IEC 13818-1 from Table 2-65 in [30] . . . 41
2.15 Box and FullBox class [12] . . . 42
2.16 Box and FullBox class . . . 43
2.17 MP3 Samples per Frame (SpF) . . . 47
2.18 MP3 Sampling Rate Frequency (Hz) . . . 47
2.19 MP3 Bit Rate (kbps) Table . . . 47
2.20 Analysis Real Sample MP2T stream duration 134s (57.7M) . . . 48
2.21 DVB-SI Tables [40] . . . 50
2.22 MPEG-2 PSI Tables [30] . . . 50
2.23 Timing DVB-SI and MPEG-2 PSI Tables [30] [40] [41] . . . 52
2.24 RTCP Packet Types . . . 57
2.25 SDES Packet Items, Identifier and Description [47] . . . 57
2.26 A sample list of RFC for RTP Payload Media Types . . . 60
2.27 RTP Header Fields meaning when RFC 2250 payload is used conveying MP2T packets . . . 61

2.28 RTP Header Fields when RFC 2250 payload is used for transporting ES streams 62
2.29 MPEG Video-specific Header from RFC 2250 [48] . . . 63
2.30 MPEG Video-specific Header Extension from RFC 2250 [48] . . . 64
2.31 Functional comparison of MMT, MP2T and RTP [46] . . . 66
2.32 HTTP Adaptive Protocols Characteristics [53] . . . 67
2.33 Comparative HLS and MS-SSTR solutions . . . 67

3.1 Example Clock Signalling at Session Level in Figure 2 from [69] . . . 75
3.2 Example Clock Signalling at Media Level. Figure 3 in [69] . . . 75
3.3 Example Clock Signalling at Sources Level. Figure 4 in [69] . . . 76
3.4 Parameters affecting Temporal Relationships within a Stream or among multiple Streams [71] . . . 77
3.5 Media Sync classification. Sync types and sub-types . . . 77
3.6 Synchronisation Methods Criteria [75] . . . 79
3.7 Synchronisation Methods Classification from [73] . . . 80
3.8 Specifications for the Colour Sub-carrier of Various Video Formats [84] . . . 83
3.9 Notation of variables in the MP2T T-STD [30] for Fig. 3.7 . . . 87
3.10 System Clock Descriptor Fields and Description [30] . . . 90
3.11 SCAR Table from [30] . . . 92
3.12 SCFR Table from [30] . . . 92
3.13 Configuration Timestamping [84] . . . 94
3.14 Film Modes States from Table 6.2 in [84] . . . 94
3.15 PTS and DTS General Calculation [84] . . . 95
3.16 Values of *PTS_DTS_flag* [30] . . . 95
3.17 Analysis of PCR values in a real MP2T sample. Analysis of number of MP2T packets between two consecutive MP2T packets containing PCRs values . . . 98
3.18 Comparison between OTR and OCR clock references . . . 100
3.19 Configuration values from SL packet, *DecoderConfig* Descriptor and *SLConfig* Descriptor when timing is conveyed through a Clock Reference Stream [33] . . . 104
3.20 Time References within ISO Base Media Format . . . 106
3.21 *stts* and *ctts* values from the track1 (video stream) from ISO example . . . 110
3.22 DT(n) and CT(n) values calculated from values in *stts* and *ctts* boxes from the track1 (video stream) from ISO example . . . 111
3.23 Descriptors for use in auxiliary Data Structure. Table 3 in [106] includes the minimum repetition rate of the descriptors . . . 123

4.1 Original video file transcoded to a MP2T format . . . 135
4.2 Original audio file MP3 format from *Catalunya Radio* (Catalonian Radio National Station) . . . 136
4.3 Description of symbols used in Fig. 4.7 . . . 140

4.4 Description of Symbols used for MP3 in Fig. 4.9 . . . 143
4.5 MP3 Frame Headers modification when positive clock skew (Delete one byte to the original MP3 frame) . . . 153
4.6 MP3 Frame Headers modification when negative clock skew (Add one byte to the original MP3 frame) . . . 153
4.7 Clock Skew Correction levels for fixed time intervals . . . 154
4.8 Clock Skew Analysis for fixed correction over adaptive time . . . 154

5.1 Analysis Formula 4 for PCR constant position within MP2T Stream . . . 164
5.2 Results Positive and Negative MP2T Clock Skew detection applied . . . 165
5.3 Audio files . . . 169
5.4 MP3 Clock Skew Detection & Correction - Effectiveness at different Skew rates . 171

1 IPTV Protocols [9] . . . 183
2 IPTV Services based on [6] . . . 184
3 IPTV Functions based on [6] . . . 185

4 SDT (Service Description Section). Table 5 in [40] (SDT Table ID: 0x42) . . . 187
5 EIT (Event Information Section). Table 7 in [40] (EIT Table ID: 0x4E) . . . 188
6 TDT (Time Date Section). Table 8 in [40] (TDT Table ID: 0x70) . . . 188
7 TOT (Time Offset Section). Table 9 in [40] with Local Time Offset Descriptor from Table 67 in [40]. (TOT Table ID: 0x73) . . . 189
8 PMT (TS Program Map Section). Table 2-28 in [30] (PMT Table ID: 0x02) . . . 190
9 PAT (Program Association Section). Table 2-25 in [30] (PAT Table ID: 0x00) . . 191

10 Clock References and timestamps main differences in MPEG standards (MPEG-1, MPEG-2 and MPEG-4) . . . 195
11 Time Fields in MPD, Period and Segment within the MPD File [59] [71] . . . 196
12 Media Delivery Techniques from [71] . . . 197

13 PMT fields with three Programs (one video and two audio) in prototype . . . 199
14 SDT with Service Descriptor in prototype . . . 200
15 PAT fields in prototype . . . 201
16 EIT fields with Short Event and Content Descriptors in prototype . . . 202
17 TDT fields in prototype . . . 203
18 TOT fields with Local Time Offset Descriptor in prototype . . . 203

19 RTP Timestamps used in prototype. Negative clock skew . . . 205
20 RTP timestamps used in prototype. Positive clock skew . . . 206
21 RTP timestamps. Negative clock skew . . . 207

22 Auxiliary Data Structure. Table 1 in [106] . . . 209

23 TVA Descriptor. Table 113 in [119]. *descriptor_tag*=0x01 210
24 Broadcast Timeline Descriptor. Table 4 in [106]. *descriptor_tag*=0x02 210
25 Time Base Mapping Descriptor. Table 7 in [106]. *descriptor_tag*=0x03 211
26 Content Labelling Descriptor. Table 2.80 in H.222 Amendment 1 [120] 212
27 Private Data structure. Table 10 in [106] 213
28 Synchronised Event Descriptor. Table 11 in [106]. *descriptor_tag*=0x05 213
29 Synchronised Event Cancel Descriptor. Table 12 in [106]. *descriptor_tag*=0x06 . 214

30 Analysis MP2T data different MP3 bitrates. Video and audio programs 216

Nomenclature

Roman Symbols

AAC Advance Audio Coded

AAL5 ATM Adaptation Layer 5

ADC Asset Delivery Characteristics

ADU Application Data Unit

AIT Application Information Table

AMP Adaptive Media Play-out

ATM Asynchronous Transfer Mode

AVI Audio Video Interleave

BAT DVB Bouquet Association Table

BCD Binary Coded Decimal

BCG Broadband Content Guide

BS Broadcast

CAT MPEG-2 Conditional Access Table

CBR Constant Bitrate

CCM System Clock Counter

CDB Compressed Data Buffer

CDN Content Delivery Network

CI Composition Information

CoD Content on Demand

CSRR Contributing Source

CTS Composition Timestamp

ctts Composition Time to Sample Box

CT UTC Clock Time

CU Composition Unit

CycCt Interleave Cycle Count

DAI DMIF Application Interface

DHCP Dynamic Host Configuration Protocol

DIT DVB Discontinuity Information Table

DLNA Digital Living Network Alliance

DMIF Delivery Multimedia Integration Framework

DSM-CC Digital Storage Media - Command and Control

DTS Decoding Timestamp

DTS Digital Theater Systems

DVB SMI DVB Storage Media Inter-operability

DVB-SI DVB Service Information Tables

DVBSTP DVB SD&S Transport Protocol

DVB Digital Video Broadcasting

DVD Digital Video Disc

e2e End-to-End

EIT DVB Event Information Table

EMM Entitlement Management Message

ESCR Elementary Stream Clock Reference

FB Feedback

FLUTE File Delivery over Unidirectional Transport

FMC FlexMux Channel

FPS Frames per Second

fps Fields per Second

ftyp File Type Box

GNSS Global Navigation Satellite Systems

GPS Global Positioning System

HbbTV Hybrid Broadcast Broadband TV

HBwTM Media Broadcast with Trick Mode

HDS HTTP Dynamic Streaming

HE-AAC High Efficiency-Advance Audio Coded

HE Head End

HNED Home Network End Device

HTC Head-end Time Clock

HTTP Hypertext Transfer Protocol

IDES Intra-Device Media Synchronisation

IDMS Inter-Destination Media Synchronisation

IETF Internet Engineering Task Force

IGMP Internet Group Management Protocol

IIS Internet Information Services

Interleave Idx Interleave Index

Internet TV TV over public unmanaged IP Networks (Internet)

IOD Initial Object Descriptor

IPMP Intellectual Property Management Protection

IPTV TV over private managed IP Networks

ISN Interleave Sequence Number

ISO BMFF ISO Base Media File Format

ITF IPTV Terminal Function

iTV Interactive TV

JD Julian Date

LMB Live Media Broadcast

LPF Low-Pass Filter

MBwTM Media Broadcast with Trick Mode

MC Multicast

mdat Media Data Box

mdia Media Box

MDU Multimedia Access Unit

mfhd Movie Fragment Header Box

MFU Media Fragment Units

MJD Modified Julian Date

MKA Matroska Audio

MKV Matroska Video

MMT MPEG Media Transport

moof Movie Fragment Box

moov Movie Box

MP2P MPEG-2 Program Stream

MP2T MPEG-2 Transport Stream

MP3 MPEG-2 Audio Layer 3

MPA MPEG Audio

MPD Media Presentation Description

MPEG-2 PSI MPEG-2 Program Specific Information Tables

MPEG-4 SL MPEG-4 Sync Layer

MPEG-DASH MPEG Dynamic Adaptive Streaming over HTTP

MPEG Moving Picture Expert Group

MPU MMU Processing Unit

MSAS Media Synchronisation Application Server

MVC Multiview Video Coding

mvhd Movie Header Box

N-PVR Network Personal Video Recording

N-PVR Network-Personal Video Recorder

NACK Negative Acknowledge

NAT Network Address Translation

NGN Next Generation Networks

NIT DVB or MPEG-2 Network Information Table

NPT Normal Play Time

NTP Network Time Protocol

OCI Object Content Information

OCR Object Clock Reference

ODA Open Data Applications

OIPF Open IPTV Forum

OPCR Original Program Clock Reference

OTB Object Time Base

PAT MPEG-2 Program Association Table

PCR Program Clock Reference

PDU AAL5 Protocol Area Unit

PETS Picture Encoding Timestamp

PLL Phase Lock-Loop

PMT MPEG-2 Program Map Table

PoC Proof-of-Concept

PTP Precision Time Protocol

PTS Presentation Timestamp

QoE Quality of Experience

QoS Quality of Service

RDS Radio Data System

RST DVB Running Status Table

RTCP FB RTCP Feedback

RTCP RR RTCP Receiver Report

RTCP SR RTCP Sender Report

RTCP Real-Time Control Protocol

RTC Real-Time Communications

RTD Real-Time Interface Decoder

RTI Real-Time Interface

RTP RET RTP Retransmission

RTP Real-Time Protocol

RTSP Real-Time Streaming Protocol

RTT Round Trip Time

SAP Session Announcement Protocol

SCASR System Clock Audio Sample Rate

SCFR System Clock Frame Rate

SC Synchronisation Client

SD&S Service Discovery and Selection

SDES Synchronisation Description

SDL Syntax Description Language

SDP Session Description Protocol

SDT DVB Service Description Table

SIT DVB Selection Information Table

SLA Service Levels Agreement

SNMP Simple Network Management Protocol

SNTP Simple Network Time Protocol

SSC System Clock Counter

SSRC Synchronisation Source

STB Set Top Box

STC System Time Clock

stts Decoding Time to Sample Box

ST DVB Stuffing Table

T-STD Transport Stream System Target Decoder

TCP Transmission Control Protocol

TDT DVB Time and Date Table

tkhd Track Header Box

TLS Transport Layer Security

TLV Protocol Type Length Value

ToD Time of Day

TOT DVB Time Offset Table

track Track Box

traf Track Fragment Box

TTS Timestamped MP2T stream

TVA TV Anytime

UDP User Datagram Protocol

UE User Equipment

UPnP Universal Plug and Play

UUID Universal Unique Identifiers

VCO Voltage-Controlled Oscillator

VoD Video on Demand

VO Video Object

WAVE Waveform Audio File Format

WMA Windows Media Audio

WMSF Web-based Synchronisation Framework

XML Extensive Markup Language

Acknowledgements

This research was partly sponsored by the Irish Research Council (IRC) and SolanoTech.

Abstract

In this thesis, the focus is on multi-source, multi-platform media synchronisation on a single device. Multimedia synchronisation is a broad research area with many facets across many multimedia application types. With convergence to Everything-over-IP, there is a growing realisation and awareness of the significant potential of Time Synchronisation in enhancing the user experience of multimedia applications. Such multimedia synchronisation can provide a totally customisable experience though such new features need to meet or surpass expected user Quality of Service/Quality of Experience (QoS/QoE). Key concerns are the number of receivers and sources, where and when to apply the synchronisation and the resynchronisation techniques applied.

As a sample use case, the thesis focuses on sports events where video and audio streams of the same event, and thus logically and temporally related, are streamed from multiple sources, delivered via IP Networks, and consumed by a single end-device. The overall objective is to showcase via the design/development of a Proof-of-Concept (PoC) how new interactive, personalised services can be provided to users in media delivery systems by means of media synchronisation over any IP Network, involving multiple sources and different IP platforms.

Papers Published

0.1 Pending Submission/Acceptance

L. Beloqui Yuste, F. Boronat, M. Montagud and H. Melvin. Understanding Timelines within MPEG Standards. IEEE Surveys and Tutorials 2015. Submitted revision August 2015.

L. Beloqui Yuste and H. Melvin. MP3 Clock Skew Detection and Correction: Technique for Intra-media Synchronisation. IEEE Communication Letters 2015. Pending submission.

L. Beloqui Yuste and H. Melvin. MPEG-2 Transport Stream Clock Skew Detection Study. IEEE Communication Letters 2015. Pending submission.

0.2 Accepted

H. Melvin, L. Beloqui Yuste, P. O'Flaithearta and J. Shannon. Time Awareness for Multimedia, TAACCS Workshop, Carnegie Mellon University, Silicon Valley Campus, US. August, 2014

L. Beloqui Yuste and H. Melvin. Interactive Multi-source Media Synchronisation for HbbTV. International Conference on Intelligence in Next Generation Networks (ICIN) - Media Synchronization Workshop. Berlin, Germany. October 2012.

L. Beloqui Yuste and H. Melvin. Client-side Multi-source Media Streams Multiplexing for HbbTV 2012 IEEE International Conference on Consumer Electronics (ICCE). Berlin, Germany. September 2012

L. Beloqui Yuste and H. Melvin. A Protocol Review for IPTV and WebTV Multimedia Delivery Systems. Journal Communications 2012. Scientific Letters of the University of Žilina, Slovakia. Issue 2/2012.

L. Beloqui Yuste, S. Al-Majeed, H. Melvin and M. Fleury. Effective Synchronisation of Hybrid Broadcast and Broadband TV. 2012 IEEE International Conference on Consumer Electronics

(ICCE) Las Vegas. January 2012.

H. Melvin, P. O'Flaithearta, J. Shannon and L. Beloqui Yuste. Role of Synchronisation in the Emerging Smartgrid Infrastructure. Telecom Synchronisation Forum (ITSF) Dublin, Ireland. November 2010.

L. Beloqui Yuste and H. Melvin. Enhanced IPTV Services Through Time Synchronisation. 2010 IEEE 14th International Symposium on Consumer Electronics (ISCE) Braunschweig, Germany. June 2010.

H. Melvin, P. O'Flaithearta, J. Shannon and L. Beloqui Yuste. Synchronisation at Application Level: Potential Benefits, Challenges and Solutions. Telecom Synchronisation Forum (ITSF) Rome, Italy. November 2009.

L. Beloqui Yuste and H. Melvin. Inter-media Synchronisation for IPTV: A case study for VLC, Digital Technologies, Žilina, Slovakia. November 2009.

0.3 Other Publications

L. Beloqui Yuste and H. Melvin. Enhancing HbbTV via Time Synchronisation. Research Engineering and IT Research Day. College of Engineering & Informatics. NUI Galway. Galway, Ireland. April 2012.

L. Beloqui Yuste and H. Melvin. Time and Timing in Multimedia. Research Engineering and IT Research Day. College of Engineering & Informatics. NUI Galway, Galway, Ireland. April 2011.

L. Beloqui Yuste and H. Melvin. Time and Timing in MPEG. IT Seminar Series, NUI Galway. Galway, Ireland. November 2010.

L. Beloqui Yuste and H. Melvin. Enhanced IPTV Services through Time Synchronisation. Research ECI-MRI Research Day. College of Engineering & Informatics. NUI Galway, Galway, Ireland. April 2010.

L. Beloqui Yuste and H. Melvin. Inter-media Synchronisation for IPTV: A case study for VLC. IT Seminar Series, NUI Galway. Galway, Ireland. November 2009.

Chapter 1

Introduction

IP Networks are widely available today in the workplace and in homes and have evolved to become the most popular media delivery platforms. The ever-evolving Next Generation Networks (NGN), which are IP based, facilitate the increase of services delivered to clients. NGN provides the media delivery platform but it would not have been possible to deliver such services without a similar evolution in media compression and delivery. The digitisation and compression technologies have thus facilitated the media delivery over any topology of IP Networks.

In this thesis, the focus is on multi-source, multi-platform media synchronisation on a single device. As a sample use case, it focuses on sports events where video and audio streams of the same event are streamed from multiple sources, delivered via IP Networks, and consumed by a single end-device. It aims to showcase how new interactive, personalised services can be provided to users in media delivery systems by means of media synchronisation over any IP Network, involving multiple sources and different IP platforms.

This raises a number of challenges and technology choices, all of which are discussed. Firstly, the media delivery platform; TV over IP Network (IPTV) and Internet TV; secondly, multimedia synchronisation; intra and inter as well as multi-source synchronisation, and finally, the technology platform used to receive and deliver the new personalised service to final users. Each are now briefly described.

1.1 IP Network Media Delivery Platform

1.1.1 IPTV

IPTV, created in 1995, is a totally different platform from traditional satellite, cable or terrestrial TV. In place of traditional broadcast technology, IPTV uses multicast IP Network media delivery. To compete with traditional systems, IPTV avoids public IP Networks and uses a private network to stream secured, copyrighted content while providing end users with the required quality. IPTV is thus, usually geographically limited to the area of influence of the

private IP Network used by the IPTV Company. Due to the geographical restriction of the distribution rights, TV Companies have to guarantee that only authorised users are entitled to access the media content.

1.1.2 Internet TV/Radio

The main advantage of Internet media delivery is to provide world wide access to media, much of which is free to Internet users. Therefore, Internet TV/Radio delivers a service through which anyone can access content from anywhere in the world only limited by copyright issues and other commercial decisions. As an example, National Catalan TV streams all the programs produced by themselves, only blocking the signal for sports events and external programs that are copyright restricted, although National Catalan Radio, as any Spanish Internet Radio, is always available worldwide. This service is especially of interest to people living abroad because, together with the Internet newspapers, it provides, a link with the media of their home country.

Internet TV/Radio utilises a variety of protocols that are distinct from IPTV. Some companies, such as Microsoft, Apple and Acrobat have developed their own proprietary Adaptive HTTP Streaming solutions for video. Dynamic Adaptive Streaming over HTTP (MPEG-DASH) is the new independent industry attempt, that aims to standardise Adaptive HTTP Streaming. In contrast to IPTV, where real time requirements can be more stringent, Internet TV/Radio can avail of HTTP which both provides reliability (via TCP) and avoids problems caused by firewalls and Network Address Translation (NAT). Adaptive techniques are designed to stream to clients, whilst adapting to different conditions such as bandwidth, bitrate, screen resolution and receiving device.

Internet Radio provides an unlimited audio choice for users. In the context of the prototype, designed and developed within this research, it gives users the choice to select their preferred audio stream to align with the video of a sporting event.

1.1.3 HbbTV

HbbTV, defined as Hybrid Broadcast Broadband TV, emerged in early 2009. Essentially, it defines the standards and the architecture that enable a receiver to access both broadcast TV and Internet media on a single device. The broadcast media delivery follows Digital Video Broadcasting (DVB) standards whereas Internet media is delivered via streaming technologies such as MPEG DASH. HbbTV, also known by the commercial term of SmartTV, is the tool that provides end-users with full interactivity with the TV delivery companies.

The concepts behind HbbTV aligns well with the research presented here in that they aim to increase end-user's personalised media services in a real-world scenario.

1.2 Multimedia Synchronisation Research

HbbTV aims to bring together different streams onto a single end-device. This PhD research aims to take this a step further by aligning or synchronising streams that are both logically and temporally related. Synchronising these streams is a significant challenge. This involves agreeing a common time standard across media sources, identifying the degree of alignment or synchronisation required, and then, the means of establishing and maintaining synchronisation.

Multimedia synchronisation is a research area comprised of several topics and which applies to many multimedia applications types. Key concerns are the number of receivers and sources, where and when to apply the synchronisation and the resynchronisation techniques applied. The synchronisation level required in each scenario can also differ greatly.

1.3 Research Motivation

Content related media is the principal context in which multimedia synchronisation is required. The most relevant example is a sports events. Films are often synchronised with other audio languages or subtitles but in this case the media streams involved would be streamed by the same source. For example, IPTV companies provide this feature whereby users can select a film language for audio or subtitles.

Synchronising multiple media streams over IP Networks from disparate sources opens up a wide range of new features. One example would be to watch a sports event from one provider whilst listening to the audio stream of the same event from a different provider. Another example could be two different sports events but in the same championship playing at the same time as the result of one game often has repercussions for the other. Users may want to have a mosaic on the screen where both games are shown simultaneously.

Users/consumers nowadays increasingly are seeking greater customisation. Such multimedia synchronisation provides a totally customised TV experience in the play-out of sports events.

Providing such new features to users is of little benefit unless the new services can meet or surpass expected user Quality of Service/Quality of Experience (QoS/QoE). Regarding the extent of required media synchronisation, significant research has been done on the thresholds for user detectability/acceptability for certain applications such as lip-sync. Such research indicates that two challenges must be met, initial media alignment/synchronisation and the subsequent detection/compensation of/for clock skew. For the live sporting application scenario presented above where synchronisation of video and separate audio commentary is required, the extend of required synchronisation is less stringent than traditional lip-sync.

1.4 Research Questions

Media synchronisation from multiple sources at client-side has to overcome a range of challenges, media timestamped at source to a common timescale, media delivered via different delivery platforms and transport protocols and media packetised via different media standards. Collectively, these impact on the synchronisation and multiplexing of the media streams into a single media stream.

Firstly, re timestamping, multiple media content servers need to be adequately synchronised otherwise, the timing process in packetising the media prior to streaming will be affected. Secondly, if media is streamed via two different media platforms, network issues, such as network jitter and network delay, could be different for each media streams. Thirdly, the choice and impact of different media transport protocols needs to be understood and addressed at client-side. Finally, each media type could use different media containers therefore different timelines need to be considered/reconstructed at client-side for synchronised integration.

These challenges represent the main research questions addressed by the thesis. They encompass the full life cycle from content production, to transport and consumption. More specifically they relate to media sources, encoding standards, and delivery platforms, and are expressed as follows:

1. Given the variety of current and evolving media standards, and the extent to which timestamps are impacted by clock inaccuracies, how can media synchronisation and mapping of timestamps be achieved?

2. Presuming that a mapping between media can be achieved, what impact will different transport protocols and delivery platforms have on the final synchronisation requirement?

3. What are the principal technical feasibility challenges to implementing a system that can deliver multi-source, multi-platform synchronisation on a single device?

Regarding content production, encoding, and timestamping, a key challenge is that all real clocks suffer from clock offset and clock skew issues. For every media streamer there are most likely two clocks involved, the server's clock and the media clock, therefore a mapping between the two may be necessary.

As multimedia encompasses a wide range of types, such as video, audio, subtitles, and other metadata, a deep knowledge of the media timelines for each is required to be able to synchronise the different media types at client-side. Moreover, the media types may have an impact on the play-out of the synchronised media at client-side, e.g., video-audio, video-metadata, video/video, and thus require different techniques to achieve a unified synchronised play-out.

Regarding delivery, the media could either be delivered via a private, well-managed IP network where QoS is guaranteed or via a free non-managed best-effort IP network such as the

Internet. The different type of networks impacts on the media delivery at the user-side and therefore has an effect on the media synchronisation at client-side.

1.5 Solution approach

The solution proposed is based on the use of existing media transport protocols along with time synchronisation protocols. These include RTP, RTCP SR as transport protocols, NTP for synchronisation timestamping, along with MPEG standards for IPTV and Internet Radio, all integrated as part of the case study. This combination of protocols facilitates, both the initial synchronisation of the media stream, and the continuous clock skew detection and consequent clock skew correction. Previous research at NUI Galway developed a mechanism for use of RTCP for skew detection across multiple sources and is protected by US patent (US 20070116057 A1). System and method for determining clock skew in packet-based telephony session.

The combination of RTP and RTCP, when implemented correctly according to standards, and when used with media sources that are synchronised via NTP, provide all the information for receivers to synchronise (both initially and via skew detection/compensation) multiple media streams sent by different media servers.

1.6 Thesis Scope

The scenario whereby user experience is greatly enhanced by the ability to present synchronised media streams from disparate sources has wide application. The particular scenario chosen is a live transmission of a sporting event. Such synchronisation benefits only arise where content is both logically and temporally related.

Multimedia synchronisation can be applied to multiple video or audio streams delivered via broadcast/broadband satellite, terrestrial, cable, IPTV. As such, multiple media containers can be used. IPTV and DVB systems typically employ MP2T. In Internet TV other media containers can be found such as Audio Video Interleave (AVI) or Matroska video (MKV) for video and MP3, Advanced Audio Codec (AAC) or Matroska Audio (MKA) for audio.

The particular case study presented here involves synchronising one video stream from IPTV, where the TV Company has the transmission rights, and another Internet Radio stream to provide an audio choice to users. As it is intended for live sporting events, subtitle streams are not considered.

RTP/RTCP is used as a common protocol to facilitate synchronisation of different media streams delivered from multiple sources. MPEG-DASH protocol is the media delivery protocol in HbbTV standards for Internet Radio and more generally Adaptive HTTP Streaming is used by all media delivered over Internet where real time delivery is not required.

Regarding scope of the thesis, the media container used for the video is MP2T, as it is used by the DVB Standard for broadcast systems. The audio container used is MP3 due to its

popularity in this scenario.

The option to listen to audio from the Internet Radio (perhaps in a different language from a different country) while simultaneously watching the sport event was chosen as it is considered to be the most likely/common use of the technology.

1.7 Contribution of this thesis

Whilst the scope of the thesis prototype is narrow in terms of use case, the overall thesis covers a much broader picture. It includes a detailed examination of a wide range of media encoding and delivery protocols involved in multicast media delivery with a special focus on synchronisation-related aspects and challenges. Having dealt with the broader topics, it then describes the design and development of a prototype to showcase multimedia synchronisation challenges and a potential solution.

The proof-of-concept (PoC) prototype implements the initial synchronisation of two media streams delivered from different sources and implements the skew detection and compensation to ensure that precise media alignment is maintained. This involves resolving for relative skew between the RTP/MP3 for audio and RTP/MP2T for video and compensating via manipulation of the audio stream. It is presumed that the sources have access to, and have implemented, a common time standard such as NTP. This is a valid presumption, as the availability of synchronised time has greatly increased in recent years due to the wider availability of precision time sources, largely through Global Navigation Satellite Systems such as GPS.

In terms of contribution, the thesis also adds to the growing realisation and awareness of the significant potential of Time Synchronisation. This is reflected in the recent US-based TAACCS [1] initiative, namely Time Aware Applications, Computers, and Communications Systems. There are strong links between the PEL Research Group at NUI, Galway and TAACCS initiative.

1.8 Thesis outline

Chapter 2 firstly distinguishes between QoS and QoE and then provides an overview of IPTV, Internet TV and the more recent development HbbTV. IPTV and Internet TV are IP Network media delivery platforms, whereas HbbTV implements a unifying media receiver at a unique end user-device. Secondly, Media Containers are explained. Finally, the media deliver protocol, RTP, used in IPTV media delivery is described.

In Chapter 3 the broad area of multimedia synchronisation is described in detail. This includes a review of recent work, very much related to the core thesis contribution such as the European project HBB-NEXT.

The proof-of-concept prototype description is found in Chapter 4 whereas the testing performed by the prototype and obtained results are described in Chapter 5. Finally, in Chapter

6, conclusions are drawn, limitations of the research are described and potential future work are presented.

Chapter 2

Media Delivery Platform, Media Containers and Transport Protocols

This chapter describes much of the foundation material for the thesis. Ultimately, the thesis proposes new techniques to improve the user experience and thus the chapter focuses firstly on the related topics of Quality of Service (QoS) and Quality of Experience (QoE). Ultimately the thesis examines the potential of synchronisation in enhancing user experience of multimedia. As such, it is important to clarify these related terms.

Having done that, the chapter proceeds with a detailed review of the fundamental components required to deliver this enhanced QoS/QoE. To consider multimedia sync at client-side from multiple sources, it is important to consider three core areas: firstly, the IP network delivery platform, IPTV or Internet TV, secondly, the media containers that deal with timelines in a different way, and finally, the protocol used for media delivery. Each protocol provides different tools which can be used for the multimedia synchronisation at receiver-side.

Regarding the first of these, the chapter examines the IP media platforms of most relevance to the thesis. For IPTV, it covers areas such as the IPTV media content, functions and services, and provides an introduction to the communication protocols used by IPTV. In Appendix A a list of the IPTV Services, Functions and Protocols is found. This section also describes Internet TV, including the codecs, containers and delivery technologies. Proprietary streaming technologies developed by software companies such as Microsoft, Apple, and Adobe Acrobat are described along with the latest MPEG Standard MMT. Finally this section presents the main HbbTV structure, media formats, and protocols used, in particular Real-Time Streaming Protocol (RTSP) (protocol for control of media delivery) and Session Description Protocol (SDP) (protocol for media session transmission).

The chapter then proceeds with a detailed analysis of the main media containers used in IPTV and Internet TV. MPEG standards are a group of documents that specify coding and packetising of media data at source for further delivery over different platforms to end-users. Whilst the section is broad in its scope, the relevant sections to the thesis implementation and proof-of-concept prototype are MPEG-2 part 1, MP3, DVB-SI and MPEG-2 PSI. As such the subsections covering the areas MPEG-4 part 1, ISO, MPEG-DASH and MMT are described to provide a general view of the different media containers in MPEG standards are not required for the specific proof-of-concept implementation. MPEG-1 was the initial standard that focused on media storage distributed in three parts: Systems, Video and Audio. MPEG-2 has more parts but the main ones are common with MPEG-1, i.e., Part 1: Systems, Part 2: Video, and Part 3: Audio. MPEG-2 Systems also included Transports Streams (MP2T) for media transmission purposes, and Program Streams (MP2P), for storage. MPEG-2 Systems also describes the specifications to packetise MPEG-1 and MPEG-4 media streams within the MP2T streams. These are all discussed in following sections.

The chapter also elaborates on the aforementioned media containers by detailing the RTP protocol used as the main media transport protocol for the media delivery. It describes RTP focusing on, the RTP timestamps, the principal RTP payload types used for MPEG-1/MPEG-2 (RFC 2250). Finally, in Appendix A, it describes RTP Retransmission (RTP RET), defined in HbbTV, and discusses issues relating to the use of RTP over UDP with NAT and Firewalls.

It is important to note that with IPTV, RTP is not obligatory, although it is recommended, whereas, for Internet media delivery, Adaptive HTTP Streaming is the predominant protocol. However, in order to more easily facilitate the synchronisation requirements, RTP with RTCP is also used for Internet audio/video delivery in the prototype.

2.1 QoS/QoE

These are two related concepts that lie at the heart of this thesis. The whole purpose of this research is to investigate the extent to which synchronised time/timing in multimedia can offer enhanced services to the end-user. Quality of Service (QoS) and Quality of Experience (QoE), although closely related, are different concepts.

QoS is defined as "*[The] Totality of characteristics of a the technical system that bear in its ability to satisfy stated and implied needs of the user of the service*" [2] whereas QoE is defined as "*the degree of delight or annoyance of the user of an application or service. It results from the fulfilment of his or her expectations with respect to the utility and/or enjoyment of the application or service in the light of the user's personality and current state*" [3].

There are three main differences between QoS and QoE; scope, focus and the assessment methods. QoS mainly focuses on telecommunication services and the measurable aspects of physical systems and thus, the analytic methods are very technology-oriented. QoE scope, on the other hand, is much wider and it is based on the user's overall assessment of the system

performance that needs a multi-disciplinary and multi-methodological approach [3].

The overall objective in the proof-of-concept is to synchronise the play-out of logically and temporally related media from separate sources. The extent to which this sync needs to be achieved is very much application dependent and has been the subject of much research over the years. The degree to which synchronisation is achieved can be technically analysed and measured, and thus is more related to QoS. Other aspects of the proof-of-concept examine the skew correction strategies deployed for MP3 and the multiplexing strategies for audio/video which although less defined are nonetheless system characteristics.

QoS and QoE expectations are very different when talking about the Internet, a free unmanaged IP network, versus IPTV, which is a well-managed service over IP Networks. Users have higher expectations when they pay for a TV services whereas are less demanding about free services delivered over the Internet. The key differences between the two IP based media delivery are explained in the following sections.

2.2 IP Network Platform

2.2.1 IPTV

IPTV offers another DVB media delivery system in addition to the traditional broadcast DVB delivery platforms, terrestrial (DVB-T), cable (DVB-C), and satellite (DVB-S). All of them are characterised by the use of different delivery platforms. The key difference with DVB-IPTV is the multicast media delivery of the content due to the underlying IP Network topology.

Traditional DVB systems use broadcast delivery meaning all channels are sent to all end-users or/and User Equipment (UE) and only one is selected for the play-out. Multicast differs because users only receive the media service selected and the IPTV media content is replicated somewhere along the IP Network.

Due to the duplex characteristic of the IP Network, IPTV services can also be interactive. Companies collect information about user behaviour and preferences, adding extra value to the IPTV Services because the information is used to increase their services by providing personalised features or advertising.

The Open IPTV Forum (OIPF) differentiates between IPTV delivered via managed or unmanaged networks. Unmanaged networks relate to media delivered via Internet where the media could be delivered by any Service Provider [4].

In this thesis, the distinction is made between IPTV, a subscription service, logically restricted and delivered via a private managed network and Internet TV, a free media delivery over Internet with no geographical restrictions.

IPTV and Internet TV, though both called broadband TV, having in common the media delivery over IP Networks, are differentiated by these key differences. Further details on distinctions are listed in Table 2.1.

	Internet TV	IPTV
Hardware	Phone/Tablet/PC/HbbTV	TV and STB/HbbTV
Software	Browser based	Media Player
	HTTP media selection	EPG
Network	Multiple Protocols - TCP based	RTP - UDP based
	Public	Private
	Unmanaged	Managed
	Worldwide access	Geographical restricted
	Mainly unicast	Mainly multicast
	Best effort service	QoS guaranteed
Media	Unprotected	Protected via encryption and security protocols
	Multiple coding	SDTV/HDTV
Media Delivery	Access to all Internet Media	Limited to IPTV content
	Not Real-Time (HTTP/ TCP)	Real-Time (RTP/UDP)
User	High Level Involvement - Lean Forward	Low Level Involvement - Lean Back
	Unsafe: Unknown users	Safe: Known users
	Free Access	Only access to known users
	Free Service	Paid Service

Table 2.1: Differences between IPTV and Internet TV

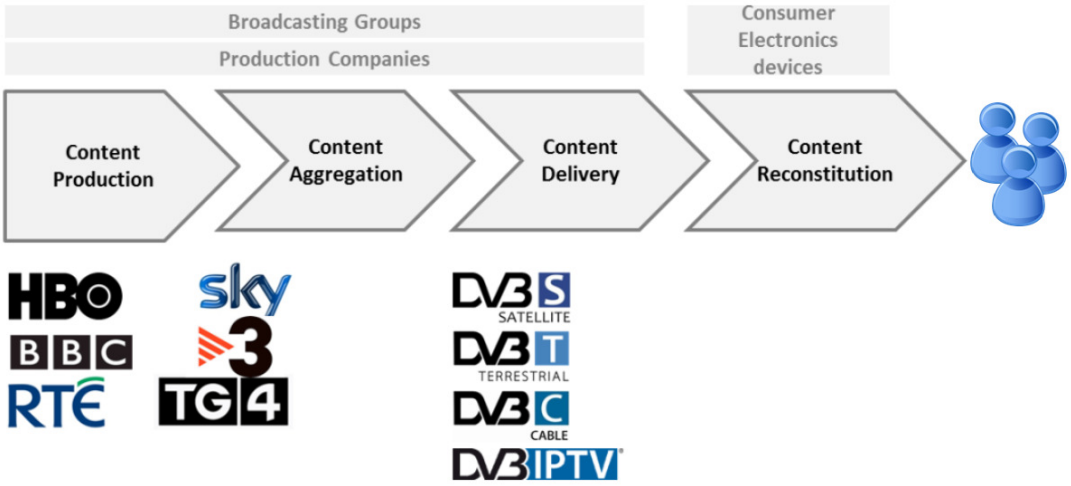


Figure 2.1: Media Content value chain in OIPF [4]

2.2.1.1 IPTV Media Content

The IPTV Media Content chain that delivers media content to end-users follows several steps: Content Production, Content Aggregation, Content Delivery and Content Reconstitution as described in Fig. 2.1.

The Content Production is the first step in the chain, it creates and produces the media content. There are multiple programs categories such as films, TV series, reality shows, news or

sports events. Second step is the Content Aggregation which groups the content into channels or group of channels, called bouquets, ready for delivery. The Content Delivery delivers the media content to end users. Finally, the Content Reconstitution is performed by the UE device on client side, such as a TV with Set-Top Box (STB), HbbTV device, PC or a mobile device [4].

Over time, many companies have played multiple roles. As example, Sky may produce a film, which once added to its catalogue, can be delivered to end-users. At the same time, Sky may sell the film's rights to other content aggregators. Another example is found in the BBC which produces most of its own programs and creates a bouquet, BBC1, BBC2, BBC3, BBC World, etc. BBC transmits its own bouquet and, simultaneously, has an agreement with Sky to deliver it via Satellite to end-users. Finally, Netflix, the Internet media streaming company, has become a producer creating its own TV shows in 2013 such as *House of Cards* and *Orange is the new Black* and providing the content delivery directly to end-users at any time via Internet TV.

2.2.1.2 IPTV Functions and Services

IPTV platforms provide a comprehensive list of services to end-users detailed in Appendix A [6].

Users typically pay a monthly subscription fee and expect to receive as many services as possible at a defined quality.

The full duplex character of IP Networks facilitates some additional services such as interactivity and personalised services, referred as Interactive TV (iTV) [7].

A user's profile can be used to generate a personalised content-guide and to provide suggestions. User's profile can also be used by IPTV companies to personalise the adverts inserted in the media content.

There is an interesting social point of view related to iTV in which the effects on social interaction are considered. One result of the full deployment of personalised TV is that the changes of different people watching the same program on the same day will greatly diminish. As a result, the social interactive discussion with other users about the program content won't take place [7].

2.2.1.3 IPTV Main Structure

There are three main roles involved in the delivery of IPTV services. These are, firstly, the Service Control Function (SCF), secondly, the Media Control Function (MCF) and thirdly, the Media Delivery Function (MDF). In Fig. 2.2 the main areas of the functional IPTV architecture services are highlighted, these are IPTV Service Controls, Transport Control, Transport Processing and IPTV Media Functions [5].

The Application and IPTV Service Control Functions performs authorization and identification, and therefore, facilitates the personalisation of the IPTV services. The Transport

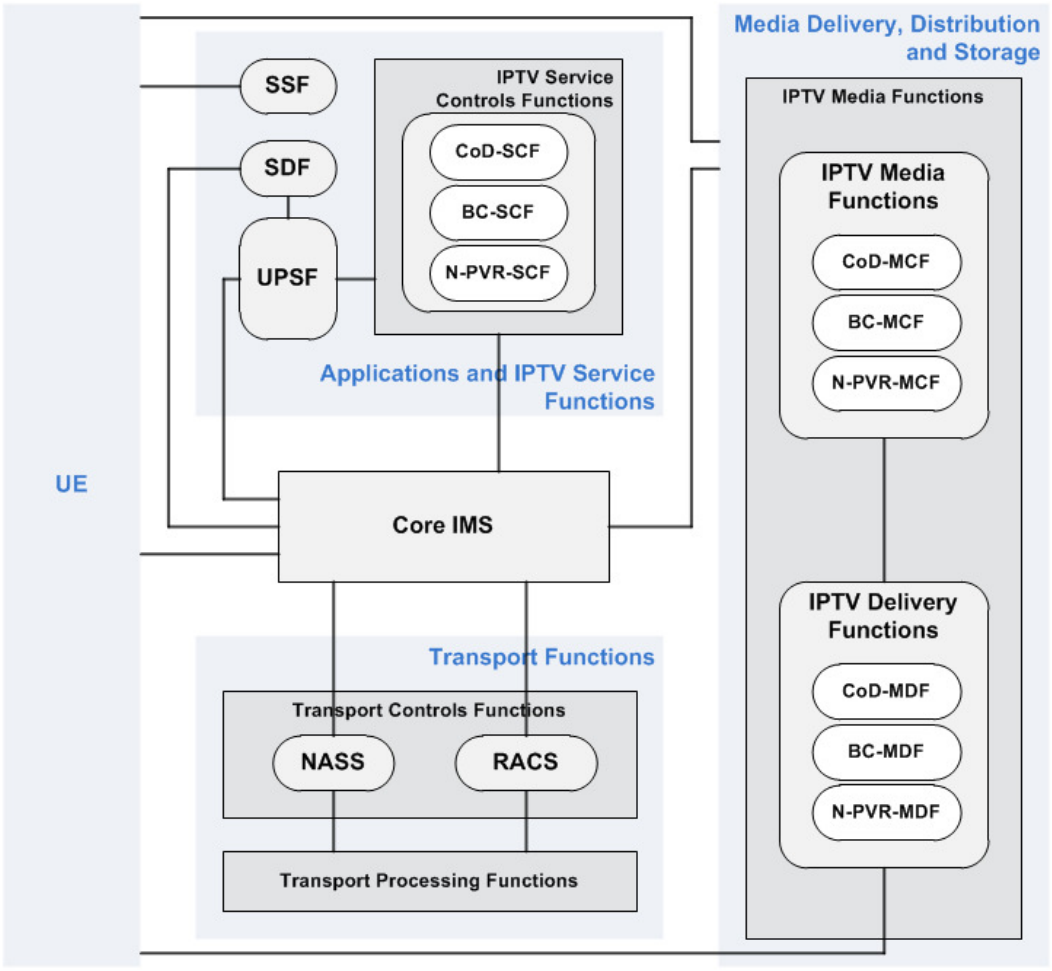


Figure 2.2: Functional architecture for IPTV Services in OIPF [5]

Functions integrates the Processing and Transport Control. The IPTV Media Functions (Media Delivery and Distribution and Storage) tasks controls and delivers the media to the UE. Inside each of the three main modules a group of sub-modules can be found where each sub-module performs a specific function. In Fig. 2.2 the sub-modules, are highlighted in light grey which are Content-on-Demand (CoD), Broadcast (BC) and Network-Personal Video Recording (N-PVR). In the following sub-sections a brief description of the sub-modules functions can be found.

Application and IPTV Service Functions

- Service Control Functions (SCF): Service authorization, credit limit and credit control of user's profile during the IPTV session initiation.

  - CoD-SCF: Content on Demand
  - BC-SCF: Broadcast
  - N-PVR-SCF: Network-Personal Video Recorder
- Service Selection Function (SSF): It provides to users the catalogue of available services. Those services can be either personalised or non-personalised. Personalized services are delivered via unicast whereas non-personalised services can be either delivered via multicast or unicast.
- Service Discovery Function (SDF): Facilitates personalised service discovery by providing the service attachment information.
- User Profile Server Function (UPSF): Stores the IMS user profile and the IPTV profile information.

Transport Functions

- Transport Processing Functions: Provides Network access links and IP core delivery data required by QoS support as a part of the IP Core.
- Transport Control Functions:
  - Resource and Admission Control Subsystem (RACS): Responsible for policy control, resource reservation and admission control.
  - Network Attachment Subsystem (NASS): Responsible for IP address provisioning, network layer user authentication and access network configuration.

IPTV Media Functions (Media Delivery, Distribution and Storage)

- IPTV Media Control Functions (MCF): Firstly, this supervises and handles MDF media flow control and MDF media processing, secondly, it controls MDF status and administers interaction with UE and IPTV SCF, and finally, it identifies and reports IPTV service state to SCF.
  - CoD-SCF: Content on Demand
  - BC-SCF: Broadcast
  - N-PVR-SCF: Network-Personal Video Recorder
- IPTV Media Delivery Functions (MDF): Manages media flow delivering report status to MCF and provides storage and support of alternative streams for personalised stream composition.
  - CoD-SCF: Content on Demand
  - BC-SCF: Broadcast
  - N-PVR-SCF: Network-Personal Video Recorder

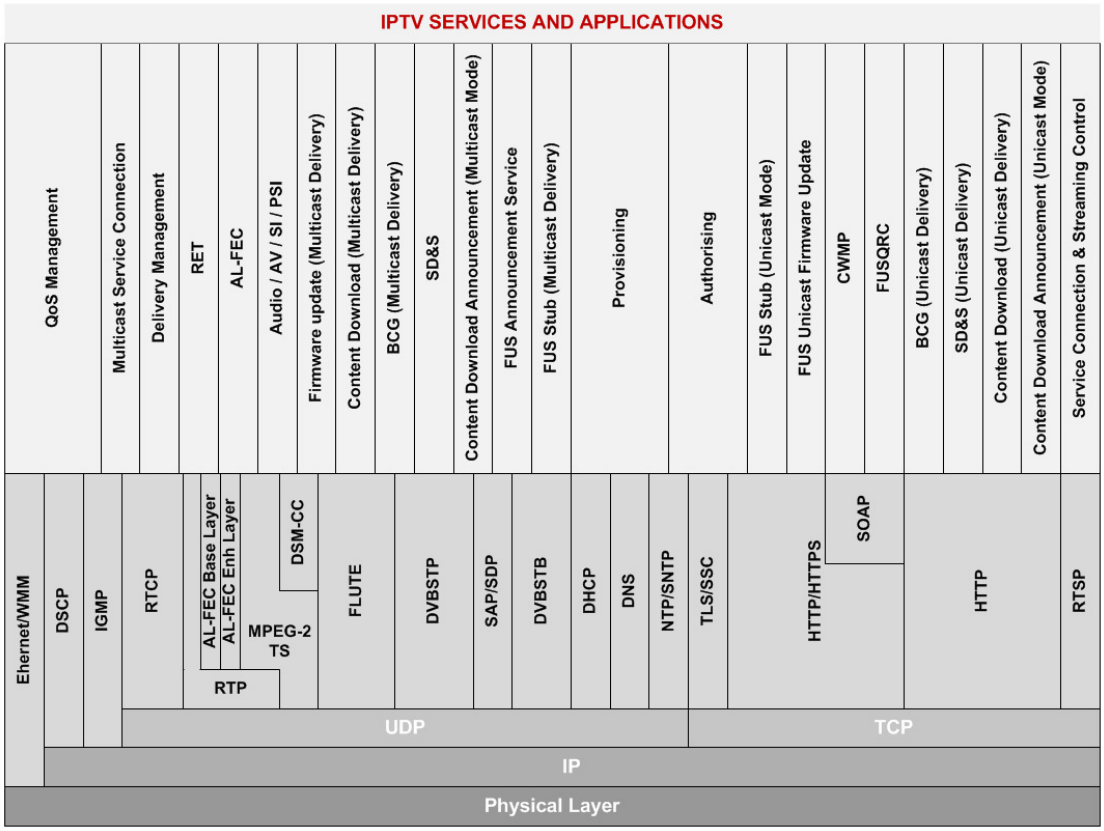


Figure 2.3: DVB-IPTV protocols stack based on ETSI TS 102 034 [8]

Core IMS Initializes the service provisioning and content delivery, facilitating the tools for authentication. Communicates with the RACS for resource reservation and admission control. Uses signalling messages to trigger the application based on the settings provided by UPSF.

User Equipment (UE) Displays information to the user to allow UE interaction, via content guides, to select broadcast or VoD services. Finally, it provides the platform for media play-out.

2.2.1.4 IPTV Communications Protocols

The overall communication process between users and the IPTV system is accomplished by the interconnection of multiple protocols. DVB-IPTV [8] and OIPF [9] define the protocol stack that provides the tools to deliver all IPTV and Internet TV services and functions to end-users. There are multiple use-cases and each of them requires different protocols between the IPTV system and end-users for different IPTV services [10]. Fig. 2.3 shows the associated protocol stack taken from [8].

Internet Group Management Protocol (IGMP) is the protocol used in multicast media de-

livery to enable users to join/leave an IPTV service. Following a service request, the Service Discovery and Selection (SD&S) is the first step in the sequence. The service selection is performed by RTSP whereby the transport mechanism of the necessary SD&S information is delivered via DVB SD&S Transport Protocol (DVBSTP) and HTTP. Once the connection is accomplished, the service is delivered and the service type will impose the protocol used at the application layer needed for its delivery [8].

Protocols such as Transport Layer Security Protocol (TLS) and Secure Sockets Layer Protocol (SSL) supply tools for authentication; DVBSTP and HTTP convey Broadband Content Guide (BCG) information to provide SD&S Service Discovery and Selection, whereas Dynamic Host Configuration Protocol (DHCP) and Domain Name System (DNS) provision the IPTV Service. Additionally, Session Announcement Protocol (SAP) and Session Description Protocol (SDP) establishes the service announcement. The media delivery uses HTTP, RTP or File Delivery over Unidirectional Transport (FLUTE) whereas Real-Time Streaming Protocol (RTSP) provides the streaming control tools to these protocols. Network Time Protocol (NTP) and Simple Network Time Protocol (SNTP) provide time synchronisation over the IP Network to all systems elements.

The media delivery protocols stream packetised media using different media containers. MPEG-2 Transport Stream (MP2T) is the media encapsulation method used to packetise the media data defined in [8]. OIPF also accepts as a media encapsulation the MP4 file format [11] and ISO Base Media File Format [12] also used by HbbTV standards when Adaptive HTTP streaming for Internet media delivery is used. Media Containers are further explained in Section 2.3.

In Fig. 2.3 the protocol stack defined by OIPF for IPTV [8] is depicted. The darkest area on the bottom of the stack corresponds to the Physical Layer. The one above is the Network Layer, mainly Internet Protocol (IP). On top of the Network Layer is found the Transport Layer mainly, i.e., UDP and TCP protocols, the choice of which is based on the protocol used at the Application Layer and the service/application needed.

Generally, UDP is used for media streaming where real-time delivery is required and TCP is used when reliable delivery is needed. RTP usually uses UDP in IPTV whereas HTTP is always used on top of TCP in Internet TV.

IGMP creates IP multicast associations, in other words, establishes multicast group memberships. This protocol facilitates end-users to join a multicast channel when media delivery is required and, finally, RTSP controls on-demand media delivery, which is described in next section of this chapter.

The most relevant protocols to this thesis will be further explained in the following chapters. A description of RTP/RTCP/RTP RET protocols (recommended although not obligatory) along with MP2T, the media encapsulation standards used in IPTV [8] are described in Section 2.3.

2.2.2 Internet TV

The concept of Internet TV as applied in this thesis relates to media delivery via Internet, that is free and geographically unlimited. The main differences with IPTV are depicted in Table 2.1. Other terminology is widely used such as Web-based TV.

Internet TV has many positive characteristics such as the free availability, geographically unlimited, stored or live media delivery, and the use of varied protocols, mainly based on firewall-friendly HTTP for its media delivery. The only drawback is the relative lack of QoS guaranteed to end users as the default service is only best effort. It must be emphasised that this is a decreasing factor due to the growing available bandwidth and increasing quality of the Internet providers and media delivery technologies. However, user applications tend to evolve to absorb available bandwidth therefore this is a never ending problem if there is no admission control. Generally speaking Internet community is happy to tolerate some occasional quality problems due to it's free access/delivery. A recent Cisco white paper published in 2014 shows the increased growth in on-line video especially consumed by mobile communication devices.

Cisco predicts a three-fold increase in VoD traffic by 2017 and that Internet Video traffic will, by 2017, represent 65% of all global IP traffic. An interesting figure is the growth of Internet Video to TV up to 34% in 2012. This last figure is especially relevant to the project since the project main idea is the play-out of a combined media stream on an HbbTV user-device. Furthermore, when mobile IP traffic only is analysed, the growth in video data is even more significant [13]. A related Cisco white paper analyses the Mobile IP traffic where again the increased usage of video delivery draws the attention. By the end of 2013, for the first time, mobile video traffic exceeded any other mobile IP traffic by a total of 53%. Cisco forecasts that in 2018, mobile video data traffic will be 69% of the total mobile traffic [14].

Internet TV, due to its global characteristics, has multiple content providers. Almost all Radio stations now stream their content via Internet, and a large number of TV companies provide free media content access either via catch up players and/or also stream in real-time. Thus, there is a large number of media codecs, media containers and media delivery systems.

Some other very popular services that come under the Internet TV classification include *YouTube* and *Netflix*. The first provides a tool to share personal videos with Internet users whereas the second provides a large choice of films and TV programs. An example of TV companies sharing their content in Internet are RTÉ with the option of watching their TV content in pseudo real-time in the Irish National Broadcaster *RTÉ Real Player* and BBC with the same service called *BBC iPlayer*.

On a related note, there are also a huge selection of Internet Radio channels. According to *Reciva* [15], there are 129 Internet Radio stations in Ireland listed in their services in April 2014 using a wide range of bits rates and formats. The majority uses MP3 format although Windows Media Audio (WMA) and Advanced Audio Coding (AAC) are also used.

Reciva provides technology to receive Internet Radio streaming without the need of a PC, laptop or a mobile device, although Reciva is also available for these devices via an application

Standard	Video	Audio	File Format
MPEG-1	MPEG-1 part 2	MPEG-1 Layer 1 (MP1) MPEG-1 Layer 2 (MP2) MPEG-1 Layer 3 (MP3)	MPEG-1 part 1
MPEG-2	H.262 part 2	MPEG-2 Layer 3 (MP3) AAC part 7	MP2T part 1 MP2P part 1
MPEG-4	H.263 part 2 H.264/AVC part 10 Web Video Coding part 29	HE-AAC part 3	ISO part 12 MP4 part 14 AVC part 15

Table 2.2: Video and Audio Codecs within MPEG Standards

or via an Internet Radio device. It supports various sampling bitrates and multiple audio codecs such as MP3, AAC, WMA, or Ogg Vorbis.

As mentioned in Chapter 2, copyright issues play an important role in the media access/delivery. As an example, with *BBC's iPlayer* for video or radio, the media is not accessible for certain media such as sports events from outside Great Britain. BBC buys the rights to transmit the sport event within a geographical area, therefore, outside this limits, UK, media content is not available.

For the project, the idea is to access a freely available Internet Radio stream of interest of an sport event and synchronise it with a restricted IPTV video of the same event.

2.2.2.1 Codecs for Internet TV

There are multiple audio and video codecs used in Internet TV, each specific to certain scenarios. Some provide better video quality, others more compression efficiency, scalability or robustness.

In Table 2.2 all the audio and video codecs in the MPEG standard are listed. One of the first was MPEG-1, part 2 for video and part 3 for audio.

Video codecs followed such as H.262 (MPEG-2 part 2), H.263 (MPEG-4 part 2), H.264/AVC (MPEG-4 part 10) and the latest Web Video coding (MPEG-4 part 29). Moreover, audio codecs follow the MPEG-2 part 3 (including the version 2 of the audio layers), and High Efficiency AAC (HE-AAC) (MPEG-4 part 3).

Table 2.3 outlines a few examples of media containers commonly used in Internet.

2.2.2.2 Media Delivery Protocols

The traditional protocol used to deliver real-time media over IP Networks, albeit not used in Internet TV, is RTP, the first protocol standardised for this use. In 1996 RTP was designed more for Real-Time Communications (RTC) such as VoIP rather than streaming and thus, for Internet TV, RTP is replaced with Adaptive Progressive HTTP Streaming techniques. In Section 2.4.1, RTP is fully described.

	Developer	File Ext	MIME Type
AVI Microsoft	Container	.avi	application/x-troff-msvideo video/avi
ASF Microsoft	Container	.asf	video/x-ms-asf
	Video	.wmv	video/x-ms-wmv
	Audio	.wma	audio/x-ms-wma
MKS Matroska	Container	.mks	
	Video	.mkv	video/x-matroska
	Audio	.mka	audio/x-matroska
OGG Xiph.org	Container	.ogg	application/ogg
	Video	.ogv	video/ogg
	Audio	.oga	audio/ogg
RM Real Media	Container	.rm	application/vnd.rn-realmedia
	Video		video/x-realvideo
	Audio		audio/x-realaudio
Flash Adobe	Container	.swf	application/x-shockwave-flash
		.flv	video/x-flv
	Video	.f4v	
	Audio	.f4a	
QuickTime Apple	Container	.mov .qt	video/quicktime

Table 2.3: Sample of Media Containers used in Internet

There are multiple streaming solutions for Internet TV but most of them are based on HTTP over TCP protocol. All of them apply Adaptive Streaming and Progressive Downloading techniques. Different software companies provide their solutions and their own protocols. Microsoft has created *Silverlight* utilizing the Microsoft Smooth Streaming Protocol (MS-SSTR) standard [16], Apple has deployed QuickTime making use of their protocol HTTP Live Streaming (HLS) [17] and, finally, Adobe Acrobat has developed Adobe Flash streaming by means of Real-Time Messaging Protocol (RTMP) [18] and the tool HTTP Dynamic Streaming (HDS) [19].

MS-SSTR, HLS, are HTTP based whereas Flash uses its own delivery protocol RTMP. More recently, Dynamic Adaptive Streaming over HTTP (MPEG-DASH), standard has been approved by HbbTV technology for Internet TV and is the independently MPEG alternative to private solutions.

Every Internet media provider selects which deployment and technology to deliver media to end-users.

For example, both Irish *RTÉ* and British *BBC*, use RTMP to deploy their on-line live player. Furthermore, the file used in the prototype is an MP3 file from the Catalan Radio Station *Catalunya Radio*, which also uses RTMP technology to deliver live radio over the Internet.

2.2.3 HbbTV

HbbTV [20] is an open platform to access services and content from multiple providers. It provides access to broadcast and broadband applications/services within a single end-user device.

A commercial name for HbbTV devices is *smartTV*.

Broadcast services support the transmission of traditional TV, radio and data services and, therefore, should support signalling, transport, synchronisation and broadcast-related applications. Moreover, broadband services (IPTV and Internet TV) provides CoD delivery, transport of related and independent broadcast applications as well as associated data.

An HbbTV end-user terminal is, thus, connected to a broadcast DVB delivery platform and to an IP Network. Therefore, HbbTV follows the OIPF and DVB specification for broadband and broadcast environments respectively to deliver interactive applications and services. The standard followed to access web-based applications at end-users devices is the CEA-2014 Standard, also called Web4CE, the Web-based Protocol and Framework for Remote User Interface on Universal Plug and Play (UPnP) Networks and the Internet [21].

Being connected to both delivery platforms, DVB broadcast and IP Network, HbbTV receives broadcast video/audio content while, via the IP Network services, also providing duplex communications channel to the TV provider. The Internet Network connection also provides pseudo real-time video/audio delivery via HTTP [22].

As depicted in Table 2.1 there are multiple differences between Internet TV and IPTV. As Internet TV is free, users don't expect such high QoS, however, with IPTV, users have higher expectations, and thus the network must be managed.

Sports video content is often transmitted via IPTV because it better facilitates live transmission, the media is protected and the companies buy the rights to the event for its users. On the other hand, multiple Internet Radio channels delivered via Internet TV are free and worldwide available even when transmitting Sport events (subject to the country copyrights policies).

Initially, IPTV was geographically limited but this is changing. For example, a Spanish Telecommunications Company, Telefónica, signed in December 2012 a contract with Ericsson to provide Telefónica's Global Video Platform, an IPTV world-wide service [23] whereas *Imagenio*, their initial IPTV Platform, is restricted to Spanish territory.

2.2.3.1 HbbTV Functional Components

In Fig. 2.4 the HbbTV Functional Components are shown. The broadcast interface receives any broadcast system Application Information Table (AIT) Data Streams event and Application Data together with Linear video/audio content. Streams Events and Application Data is conveyed via Digital Storage Media - Command and Control (DSM-CC) object carousel[1]. The DVB AIT table structure is defined in Table 2.4.

The DSM-CC Client receives the DSM-CC object carousel, Streams Events and Application Data, whereas the AIT Filter receives the DVB-SI AIT Table to filter the application information.

The broadband interface receives the AIT Data, the Application Data and the Non-Linear

[1]Data broadcast to users related to the media standard format

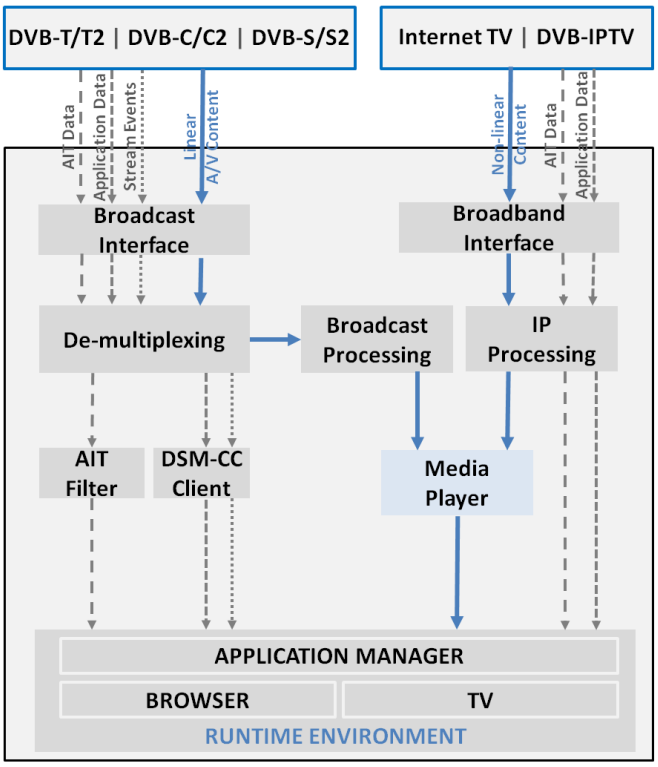


Figure 2.4: HbbTV High Level architecture. Figure 2 in [22]

video/audio data (received via IP networks) and sends it to the IP Processing block.

The Broadcast Processing module receives the Linear A/V content (broadcasted to users via DVB) which is sent to the Media Player where also the Non-Linear video/audio content is sent to Internet Protocol Processing block.

In Fig. 2.4 the DSM-CC and AIT data have grey arrows whereas the DVB Media Content is blue. The main difference is that the Broadband Interface does not receive any Streams Events data. As shown, both Linear A/V Content (DVB Media Content) and Non-linear A/V Content (IPTV and Internet TV) are sent to the HbbTV Media Player module (also shown with blue background).

In Broadcast TV Application transport and synchronisation follow DSM-CC. On the other hand MPEG-2 is used for broadcast signalling and XML is used for Broadcast Independent application signalling [24].

2.2.3.2 Formats

ETSI TS 102 796 [22] specifies the media formats which follow the OIPF Media Formats specification [25]. Here it is presented a summary of media formats in both specifications.

Field	Bits
application_information_section () {	
table_id	08
section_syntax_indicator	01
reserved_future_use	01
reserved	02
section_length	12
test_application_flag	01
application_type	15
reserved	02
version_number	05
current_next_indicator	01
section_number	08
last_section_number	08
reserved_future_use	04
common_descriptors_length	12
for (i=0; i<N; i++) {	
descriptor(){	
}	
reserved_future_use	04
application_loop_length	12
for (i=0; i<N; i++) {	
application_identifier()	
application_control_code	08
reserved_future_use	04
application_descriptors_loop_length	12
for (i=0; i<N; i++) {	
descriptor(){	
}	
}	
CRC_32	32
}	

Table 2.4: Application Information Section. Taken from Table 16 in [24]

Broadcast-specific System, video and audio format are not defined, the '*requirement are defined by the appropriate specifications for each market where terminals are to be deployed*' [22].

Broadband-specific: Systems Layers System, video and audio formats follow the OIPF Media Formats specifications [25]. In Table 2.5, the formats used are listed. TTS is named as the special MP2T format used by IEC 62481-2[1] [25]. TTS is a special MP2T media container referred as Timestamped MP2T stream (TTS) [26].

[1]Describes Digital Living Network Alliance (DLNA) media format profiles applicable to the DLNA device classes defined in IEC 62481-1

Service	Transport Protocol	Systems Layer Format
Scheduled Content	Direct UDP or RTP/UDP	MP2T, TTS
Streamed CoD[a]	Direct UDP or RTP/UDP	MP2T, TTS
Streamed CoD[b]	HTTP	MP2T, TTS, MP4
Download CoD	HTTP	MP2T, TTS, MP4

Table 2.5: Systems Layer formats for content services. Table 6 in [25]

[a] only used in IPTV
[b] used in Internet TV

Broadband-specific: Video High Definition (HD) and Standard Definition (SD) are supported. Two formats are used, H.264/AVC and MPEG-2. That means for HD it is AVC_HD_30, AVC_HD_25 and MPEG2_HD_30, and for SD it is AVC_SD_30, AVC_SD_25 and MPEG2_SD_30. Finally, the format AVC baseline profile at level 2 should be supported [25].

Broadband-specific: Audio Formats for audio include HE-AAC, ACC, AC-3, Enhanced AC-3, MPEG-1 Layer II, Layer III, Waveform Audio File Format (WAVE), Digital Theater Systems (DTS) Sound System, and MPEG Surround [25].

2.2.3.3 Protocols

In Fig. 2.5, an overview of the protocol stacks used in IP Networks in HbbTV (except MMT which is a standard recently approved in 2014) are shown.

Broadcast-specific DSM-CC and *caching_priority_descriptor* should be supported. For broadcast signalling, MPEG-2 descriptors should be supported following the specification. Moreover broadcast-independent applications if they are signalled should use AIT encoded via XML format [24].

Broadband-specific Broadband TV protocols used for media streaming are HTTP and the protocols used for unicast streaming for MPEG-4/AVC and MPEG-4/AAC are RTSP and RTP. Download functionality is facilitated by HTTP and the application transport is performed by HTTP or HTTP over Transport Layer Security (TLS) [22].

2.2.3.4 Applications

Broadcast-dependent application (IPTV) can be conveyed via a carousel explained. The two objects, streams events and application data, are conveyed via one or multiple MP2T streams.

Broadcast-independent applications (Internet TV) do not need any signalling, information is transmitted using AIT using XML delivered via HTTP. The Mime Type used for Broadcast-independent applications is *"application/vnd.dvb.ait+xml"*.

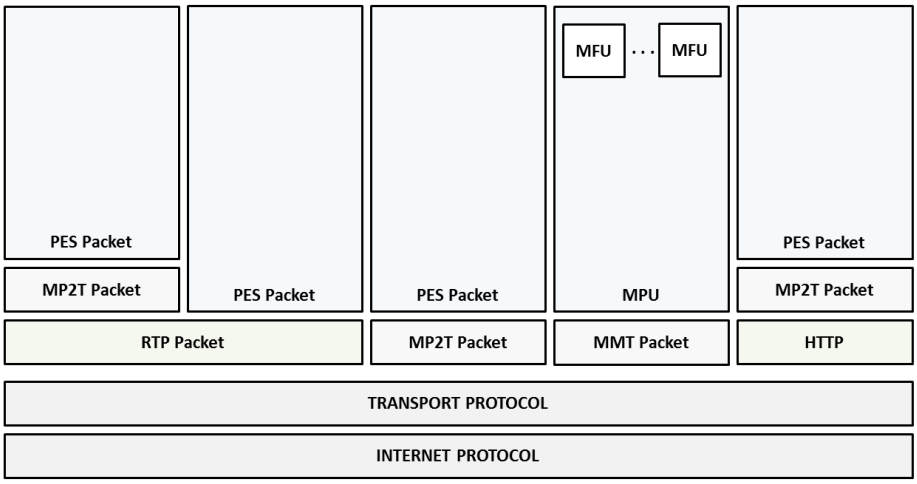


Figure 2.5: Media Delivery Protocols Stack with RTP, MPEG-DASH and MMT. Green: RTP and HTTP; grey for MP2T/MMT packet and blue PES and MPU packets

2.2.3.5 HbbTV video/audio

Linear video/audio received via broadcast, DVB-S, DVB-T or DVB-C, are delivered following the DVB MP2T. Non-Linear video/audio received via broadband is subdivided in two categories. First is DVB-IPTV which is delivered following [8] and Internet TV which is delivered via multiple protocols though mostly HTTP based using Adaptive HTTP protocols.

2.2.3.6 RTSP

RTSP is the Application Layer Protocol that facilitates the control of on-demand real-time media delivery for IPTV. It does not stream the media but it gives users the tool to control the on-demand media delivery chosen. In other words, the function is similar to a Digital Video Disc (DVD) player remote control giving users the tool to set-up, start, pause and tear-down the media play-out within a media session [27].

HTTP and RTSP functions are deployed with some differences. RTSP maintains the state of the media session where client and server can issue requests. HTTP is a stateless protocol where only the client generates request and the server responds.

Although RTSP and RTP work hand in hand in the process of final media delivery to users, they are not tied to each other. In Fig. 2.6, one example of an RTSP communications timeline including the RTP/RTCP messages within the media session is shown. Firstly, the session begins with a RTSP describe command and secondly the session is set-up via an *RTSP setup* message, *RTSP play* then starts media delivery via RTP/RTCP. RTP delivers the media content while RTCP packets provide information about the quality of the media session. It is up to the client to send a *RTSP teardown* packet to inform the RTSP server about the end of the media session.

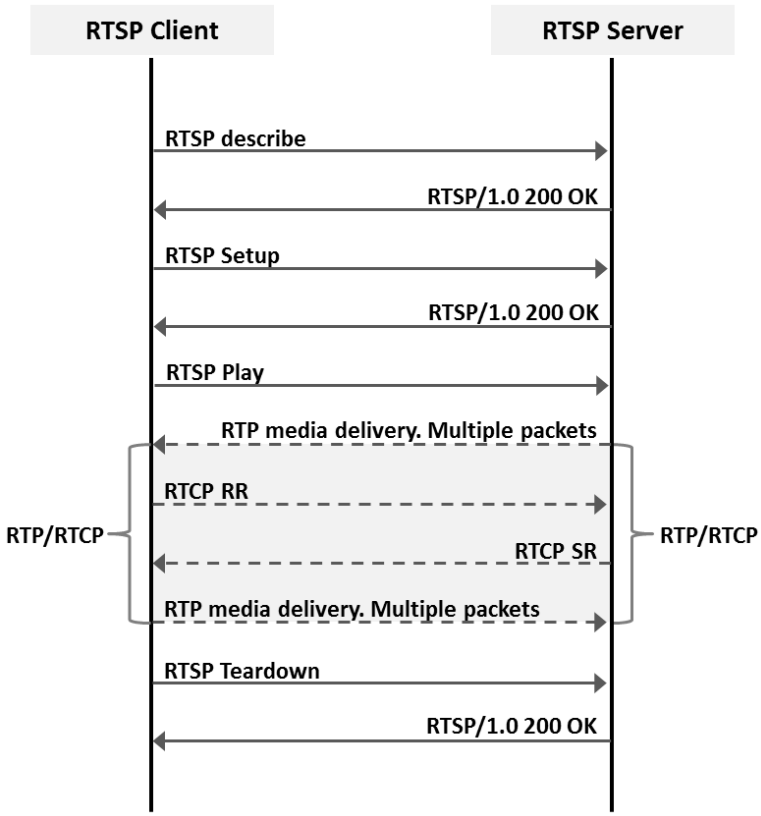


Figure 2.6: RTSP communications with RTP/RTCP media delivery example

```
npt-range     =  ( npt-time "-" [ npt-time ] ) | ( "-" npt-time )
npt-time      =  "now" | npt-sec | npt-hhmmss
npt-sec       =  1*DIGIT [ "." *DIGIT ]
npt-hhmmss    =  npt-hh ":" npt-mm ":" npt-ss [ "." *DIGIT ]
npt-hh        =  1*DIGIT     ; any positive number
npt-mm        =  1*2DIGIT    ; 0-59
npt-ss        =  1*2DIGIT    ; 0-59
```

Figure 2.7: RTSP Format Play Time [27]

RTSP functionality is based on methods that provide the control over the media delivery. Some of them such as *options*, *describe*, *announce*, *get_parameter*, *set_parameter*, *redirect*, return embedded binary data whereas methods such as *setup*, *play*, *record*, *pause*, and *teardown* alters the state of the RTSP connection [27].

With RTSP the play time and the absolute time can be transmitted to the users. The Normal Play Time (NPT) is relative to the beginning of the media play-out. Absolute time indicates the wall clock time of the media play-out. Both follow ISO 8601 Standard [28]. In Fig. 2.7, the syntax of the play time can be found, followed by Fig. 2.8 which outlines the Absolute Time syntax.

```
utc-range    =  "clock" "=" utc-time "-" [ utc-time ]
utc-time     =  utc-date "T" utc-time "Z“
utc-date     =  8DIGIT                ; < YYYYMMDD >
utc-time     =  6DIGIT [ "." fraction ]  ; < HHMMSS.fraction >
```

Figure 2.8: RTSP Absolute Time [27]

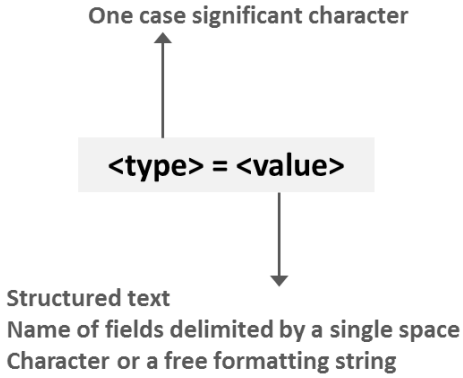


Figure 2.9: SDP Main Syntax Structure

2.2.3.7 SDP

SDP describes a multimedia conference as '*a set of two or more communicating users along with the software they are using to communicate*' [29] and a multimedia session as a '*set of multimedia senders and receivers and the data streams flowing from senders to receivers*' [29].

SDP is the protocol used to standardise the means to transmit information within the multimedia session initialization process. SDP is autonomous from the transport protocols used to stream the multimedia data, and only provides information to facilitate the communication between end2end (e2e) media sessions. A multimedia session requires standard media information, transport address and session description metadata, which is provided by SDP at the commencement and during the session.

Session Description describes the session name and purpose, session active time, the session media and any other information needed by the session receivers. Media information includes the type (audio, video, application) and the format (audio/video codecs). The transport information conveys information about the protocols used for the multimedia delivery over the network. The syntax used by SDP is described in Fig. 2.9 and all SDP parameters used are listed in Table 2.6.

Session-level description information relates to the complete session and all media streams whereas Media-level description only relates to a single media stream within the session.

Finally, two different types of IP delivery can be found, multicast and unicast. In the former, information about the multicast group address and the transport port for media distribution is required. In the latter, remote address and remote transport port for media delivery is needed. The syntax of the different description levels is as follows:

Level		Type (o=optional)		Information
Session	v	Protocol version		
	o	Originator and session identifier		
	s	Session Name		One per session description Characters ISO 10646
	i	Session Information	o	One or more per session. At least one per each media
	u	URI of Description	o	One URI per session
	e	Email Address	o	Multiple values allowed
	p	Phone Number	o	Multiple values allowed
	c	Connection Information	o	
	b	Bandwidth information lines	o	<modifier><bandwidth-value>
	z	Time Zone adjustments	o	<adjustment time><offset>
	k	Encryption Key	o	<method>:<encryption key>
	a	Session attribute lines	o	
Time	t	Time the session is active		<start time><stop time>
	r	Zero or more repeat times	o	
Media	m	Media name and transport address		
	i	Media title	o	
	c	Connection information	o	If present at session level is not need it
	b	Bandwidth information lines	o	
	k	Encryption Key	o	
	a	Media attribute lines	o	

Table 2.6: SDP parameters

- Session identifier: *o=<username><session id><version><network type><address type><address>*
- Media syntax (The media can be audio, video, text, application and message): *m=<media><port><protocol><fmt><att-field><bwtype><nettype><addrtype>*
- Connection Data: *c=<nettype><addrtype><connection-address>*
- Bandwidth: *b=<bwtype>:<bandwidth>*

In Section 3.1.2 a proposed IETF standard is described where extra information about clock signalling expands the information provided by SDP to facilitate media synchronisation, which is of particular relevance to this thesis.

2.3 Media Containers

2.3.1 MPEG-2 part 1: Systems

MPEG-2 part 1, Systems, describes the two media containers structures available in MPEG-2. These are the Program Streams (MP2P) and Transport Streams (MP2T), and each have dif-

Fields	Bits
MPEG2_program_stream () {	
do {	
pack ()	
} while (nextbits() == pack_start_code)	
MPEG_program_end_code	32
}	

Table 2.7: MPEG-2 Program Stream Structure. Table 2-31 in [30]

Fields	Bits
pack () {	
pack_header ()	
while (nextbits () == packet_start_code_prefix) {	
PES_packet ()	
}	
}	

Table 2.8: MPEG-2 Pack Structure. Table 2-32 in [30]

ferent purposes.

MP2P is designed for error-free environments such as storage and local play-out. MP2P only conveys a single program with a unique timebase. MP2T on the other hand is designed for environments where errors are common such as streaming over IP Networks or broadcasting via DVB. It conveys multiple programs each of them associated with its own timebase. Both structures, MP2P and MP2T, convey Packetised Elementary Stream (PES). The main differences about timelines between MP2P and MP2T are further explained in Chapter 3.

In Table 2.7 the main structure of a MP2P is found. Every MP2P stream has multiple packs. The MP2P finishes when the *MPEG_program_end_code* is found. Table 2.8 shows the pack's main structure. Each pack is constructed from one variable size *pack_header* and multiple PES_packets. The pack's header is depicted in Table 2.9. Finally, within the *pack_header*, the time-related field System Clock Reference (SCR) is found.

The MP2T streams follow a different structure than MP2P. It is designed for a non-free error environment and is thus of most relevance to the thesis. The packets have a fixed size (188 bytes). Every MP2T stream can convey multiple programs moreover, each program follows an independent timeline, namely Program Clock Reference (PCR), and each program can convey multiple media streams (e.g., one program can include one video stream and three audio streams), all of them linked to the PCR timeline of the related program. For example, in the prototype described in Chapter 4, one option implemented is to add a second audio stream to an existing video stream.

Fig. 2.10 represents the MP2T packet high level structure. Each packet is 188 bytes, including a 4 byte MP2T header, adaptation field and a part of a PES (including perhaps a PES

Field	Bits
pack_header () {	
pack_start_code	32
'01'	02
System_clock_reference_base [32..30]	03
marker_bit	01
System_clock_reference_base [29..15]	15
marker_bit	01
System_clock_reference_base [14..0]	15
marker_bit	01
System_clock_reference_extension	09
marker_bit	01
program_mux_rate	22
marker_bit	01
marker_bit	01
reserved	05
pack_stuffing_length	03
for (i=0; i<pack_stuffing_length; i++) {	
stuffing_byte	08
}	
if (nextbits() == system_header_start_code) {	
system_header ()	
}	
}	

Table 2.9: Pack Header Structure. Table 2-33 in [30]

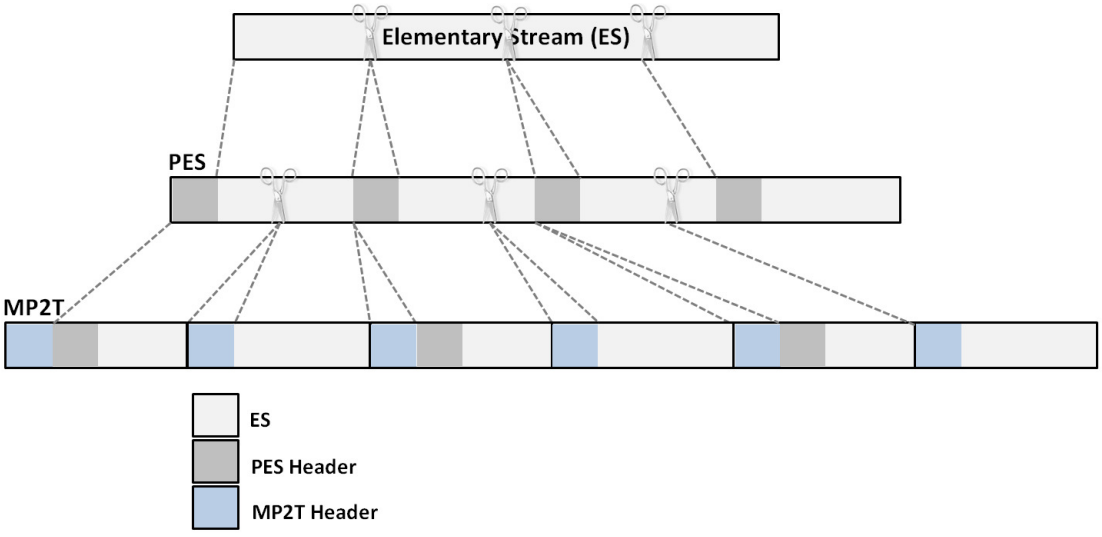


Figure 2.10: Process to packetised a PES into MP2T packets. Multiple MP2T packets are needed to convey one PES

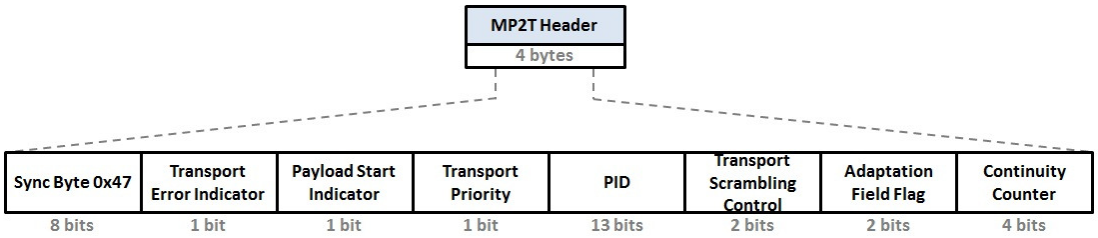


Figure 2.11: MP2T Header and fields

Field	Bits
MPEG_transport_stream () {	
do {	
transport_packet ()	
} while (nextbits() == sync_byte)	
}	

Table 2.10: MPEG-2 Transport Stream Structure. Table 2-1 in [30]

Fields	Bits
transport_packet () {	
sync_byte	08
transport_error_indicator	01
payload_unit_start_indicator	01
transport_priority	01
PID	13
transport_scrambling_control	02
adaptation_field_control	02
continuity_counter	04
if (adaptation_field_control=='10' || adaptation_field_control=='11') {	
adaptation_field ()	
}	
if (adaptation_field_control=='01' || adaptation_field_control=='11') {	
for (i=0;i<N; i++) {	
data_byte	08
}	
}	
}	

Table 2.11: MPEG-2 Transport Stream Packet Structure. Table 2-2 in [30]

header and PES payload). The MP2T header fields are shown in Fig. 2.11. The MP2T stream structure is found in Table 2.10 and the MP2T packet structure in Table 2.11. One MP2T packet conveys an 4-byte size header, *data_byte* and optionally, an *adaptation_field*, signalled by the *adaptation_field_control* field. The *data_byte*, which is essentially the MP2T payload, could contain, PES load or PES load with a PES header, DVB-SI or MPEG-2 PSI tables, auxiliary

data or data descriptors. The general MP2T structure follows Fig. 3.8a in Chapter 3.

2.3.2 MPEG-4 part 1: Systems

The MPEG-4 Systems are based on the Elementary Stream Management. An MPEG-4 elementary stream contains the encoded audio-video objects, scene description and control information. The Elementary Stream Management is the tool to describe data stream the relation between data streams which are tightly related to media synchronisation [31].

The Media Object Description Framework provides the means to describe the MPEG-4 media. The main elements are the Object Descriptor Components, the Transport Encapsulation of Object Descriptors and the information of the Usage of the Object Descriptors [32], which are explained in the following sections.

2.3.2.1 Architecture

'The information representation specified in ISO/IEC 14496 describes the means to create an interactive audio-visual scene in terms of coded audio-visual information and associated scene description information' [33].

The coded representation is sent by the encoder to a receiver where it is received and decoded. Encoder and decoder are given the general term audio-visual terminal or terminal [33]. To accomplish this process, to decode, the information received in an Initial Set-up Session (specified in 14496-6) allows the receiving terminal to access content representation conveyed in the elementary streams [33].

The terminal architecture, as seen in Fig. 2.12, begins at the transmission/storage medium, followed by the delivery, sync and compression layer. The final layer, the composition and rendering, is applied at the end-user's final terminal, either a TV set, a lap-top or any mobile device [33].

MPEG-4 Systems is based on the use of Object descriptors that provide the information about the media data, named Object Description Framework.

2.3.2.2 Terminal Model

The systems decoder model, comprised of the buffer and timing model, determinates the decoder's performance. Buffer management and synchronisation are required in order to correctly display the media streams at the receiver [33].

The timing model function is defined as *'the mechanisms through which a receiving terminal establishes a notion of time that enables it to process time-dependent events. This model also allows the receiving terminal to establish mechanisms to maintain synchronisation both across and within particular audio-visual objects as well as with user interaction events'* [33].

The buffer model function is defined as '*The buffer model enables the sending terminal to monitor and control the buffer resources that are needed to decode each elementary stream in a*

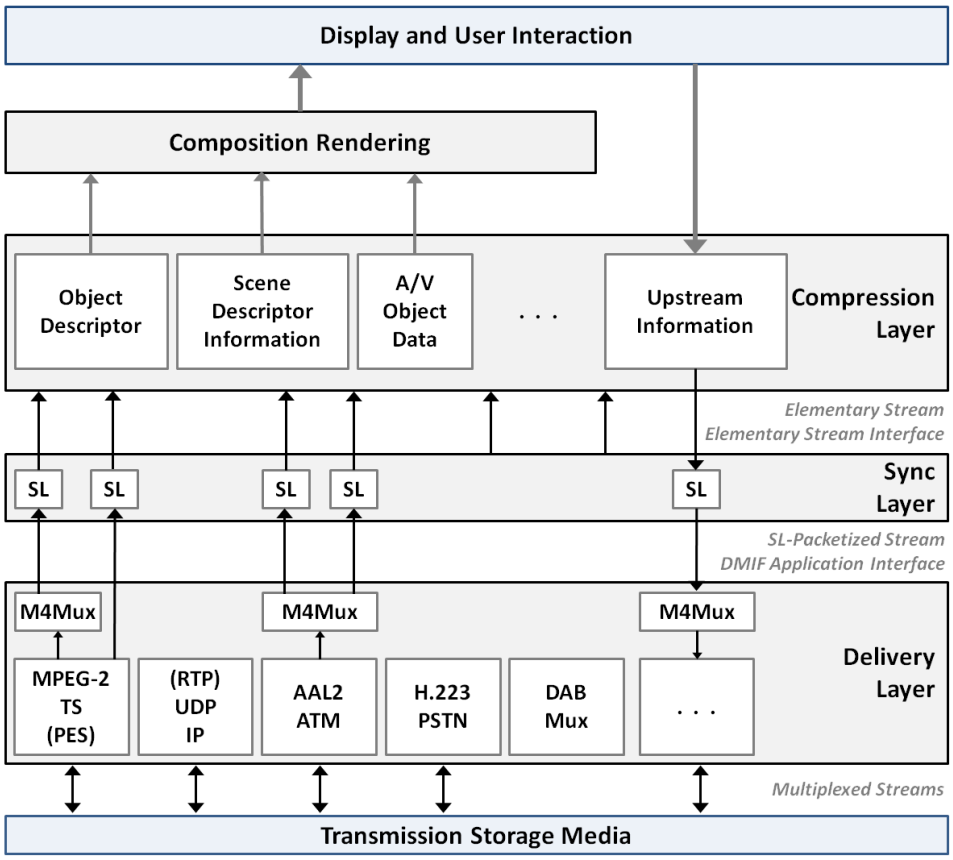


Figure 2.12: MPEG-4 Terminal Architecture. Figure 1 in [33]

presentation. The required buffer resources are conveyed to the receiving terminal by means of descriptors at the beginning of the presentation' [33].

The Terminal Architecture comprises the Delivery, Sync and Compression Layer as shown in Fig. 2.12. The Delivery Layer may involve different protocols depending on the application, the Synch Layer is based on Sync Layer packets and optional FlexMux Packets whereas the Compression Layer is formed by all descriptor structure and audio/video streams.

DMIF Application Interface (DAI), specified in 14496-6, also known as Delivery Layer in Fig. 2.12 establishes the delivering data interface and provides necessary signalling information for session/channel set-up and tear-down. Multiple delivery mechanisms, some suggested in Fig. 2.12, are found above this interface to accomplish transmission and storage of streaming data [33].

Timing at the Synch Layer in Fig. 2.12 facilitates synchronising the decoding and composition processes of the elementary streams, composed by access units (AU). Elementary streams are carried as SL-packetised streams which provide first of all timing information, second, synchronisation and random access information, and finally, fragmentation [33].

The Compression Layer in Fig. 2.12 receives the different encoded data streams being responsible for the decoding of the AU. It is the step prior to the composition, rendering and presentation to the final user. The Compression Layer utilizes the Object Description Framework to accomplish its tasks [33].

2.3.2.3 Object Description Framework

The functionality of the Object Description Framework involves defining and identifying elementary streams, their inter-connection and lastly the association with audio-visual objects used in the scene description. *ObjectDescriptorsID* is the identifier used to associate the object descriptors with the nodes within the scene description. The transport of the scene descriptors and the audio-visual data is performed by ES [33] (See Fig. 2.13).

In Fig. 2.14 the scene, which reflects what the prototype implementation described in Chapter 4 would look like if using MPEG-4, has four visual objects (background, player_1, player_2, player_3 and the ball) and two audio objects (English and Catalan audio). The Object Description Framework provides information of all the objects and how they are used within the scene. Objects can be linked to one or more streams, i.e., every object in the example is linked to two visual streams, Base and Enhancement Layer. At the same time both representations *Movie Texture A* and *Movie Texture B* have the two audio streams ES_ID so both visual representations have the two audio options available for user-choice.

The scene descriptor establishes the spatio-temporal association between audio-visual objects. The stream information is complemented by the object description framework providing information about the scene. Object Descriptors are composed of a collection of descriptors which describe the elementary streams [33]. Fig. 2.13 shows the mapping between Object and Scene Descriptors and the Media streams.

An example of BIFS (scene and object descriptors) is found in Fig. 2.14. *InitialObject Descriptor* points at the Scene and Object Descriptor Stream. The Scene Description Stream (in orange) conveys the BIFS tree structure. The Object Description Stream (in green) conveys all the object descriptors part of the BIFS node tree.

The Object Description Framework principal aim is to recognize and detail the elementary streams and link them with the correct audio-visual scene descriptor. The main components of the Object Description Framework are firstly the audio-visual streams and secondly the descriptor streams which provide the audio-visual streams information required for decoding, composition and presentation. Fig. 2.13 describes the connections between the different descriptor streams and the audio-video streams [33].

An Object Descriptor consists of multiple streams providing information about audio, video, text, or data streams. In Fig. 2.14 it should be appreciated that one object descriptor can convey the ES_ID for two video streams (one for the Base Layer and the Enhancement Layer). Object descriptors are carried in elementary streams. Identification is performed by a unique identifier (Object Descriptor ID) which is used to link object descriptors with the audio-visual

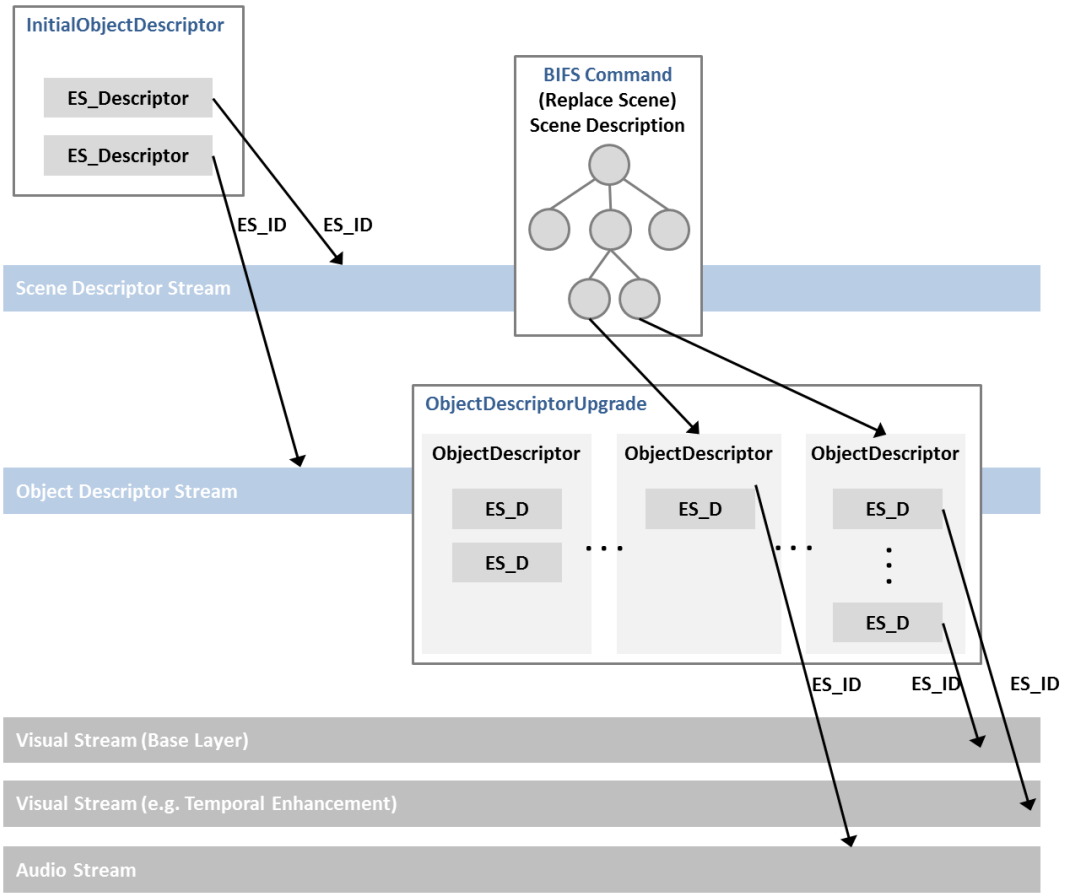


Figure 2.13: Object and Scene Descriptors mapping to media streams. Figure 5 in [33]

objects within a scene description [33].

A scene node is associated with multiple elementary streams described by an Object Descriptor which relates to a single audio or visual object. Scene descriptors manage the spatial and temporal attributes to coordinate the audio-visual objects within a scene. Scene descriptors are organized in BIFS (Binary Information Format for Scenes) and conveyed in Scene Descriptor Streams. The scene descriptor streams are organised as a tree of nodes. Leaf nodes carry the audio-visual data therefore, the intermediate nodes group the audio-visual data into audio-visual objects to perform different types of operations on the audio-visual objects [33].

Elementary Streams from source to receiver require up-channel information sent by the terminal (user) to the source (media server). Every up-channel stream is associated with a down-stream elementary stream.The user interaction information is not defined in 14496 part 1 although it is a requirement during scene rendering. Interaction information is translated into scene modifications which is also reflected into the composition process [33].

In Fig. 2.14 there are two intermediate nodes, *Movie Texture A* and *Movie Texture B* which

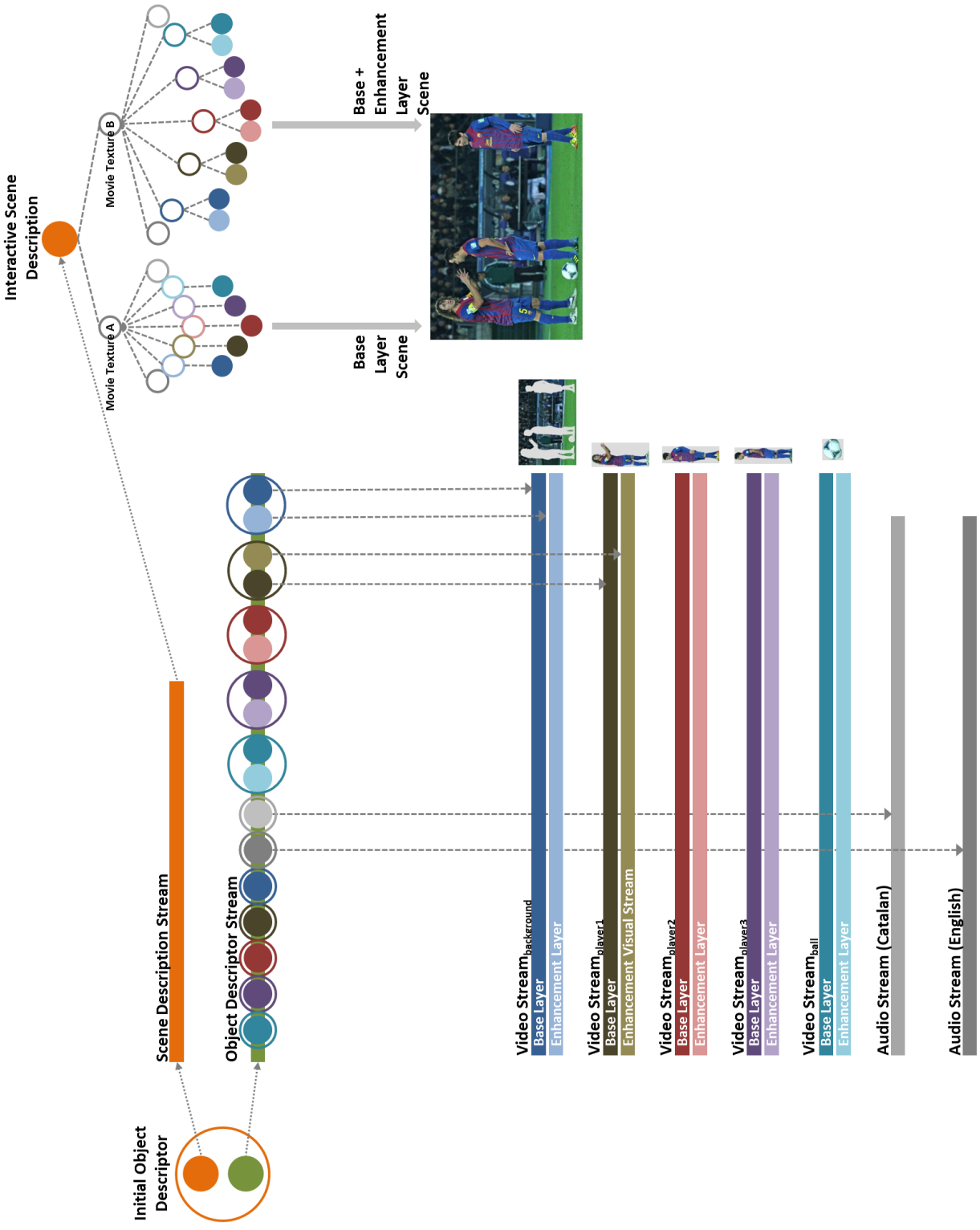


Figure 2.14: Example BIFS (Object and Scene Descriptors mapping to media streams) following example Figure 2 from http://mpeg.chiariglione.org/

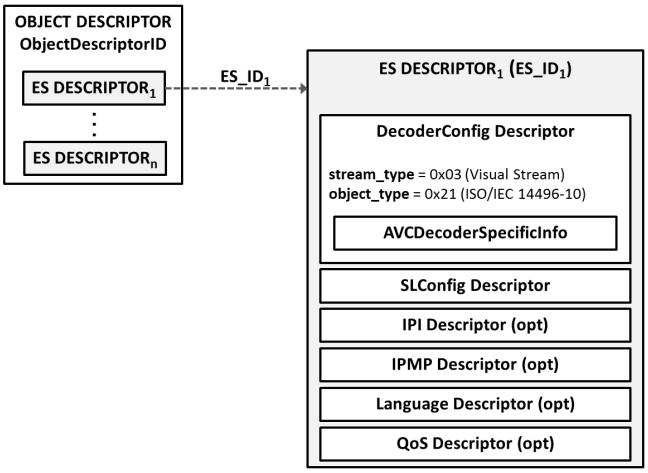


Figure 2.15: Main Object Descriptor and related ES Descriptors

are the two possible representations of the Scene. The former displays the scene only using the Base Layer while the latter uses the Base and Enhancement Layer, therefore, is better quality. The object descriptors from the *Movie Texture A* only have one ES_ID which links to the Base Layer Video streams. However, the object descriptors from the *Movie Texture B* have two ES_ID one linked to the Base Layer and the second one to the Enhancement Layer. *Movie Texture B* thus needs the two ES_ID for both visual streams to decode the video object.

The main components of the Object Descriptors are: ES, OCI (Object Content Information), IPMP (Intellectual Property Management and Protection), SL (Sync Layer), Decoder, QoS and Extension Descriptors.

ES descriptor Elementary Stream Descriptors include information used by the transmission and decoding process. Information such as the source of the data stream, encoding format, configuration information, QoS requirements and intellectual property identification. All this information is also provided with the dependencies among streams conveyed in the Elementary Stream Descriptors [33].

Fig. 2.15 illustrates an example of the Object Descriptor linked to ES Descriptors as well as an example of descriptors within (ES_ID_1). *DecoderConfig* and *SLConfig* descriptors are obligatory whereas the rest are optional.

There will be an encoder/decoder allocated to each ES. Fig. 2.16 shows the block diagram of these encoders/decoders. Each ES and the encoder used is linked via the *ES* descriptor and the *DecoderConfig* descriptor.

OCI descriptor OCI contains information about audio-visual objects in a descriptive format. Information is classified in descriptors such as content classification, keywords, rating, language, text data, creation context descriptors [33].

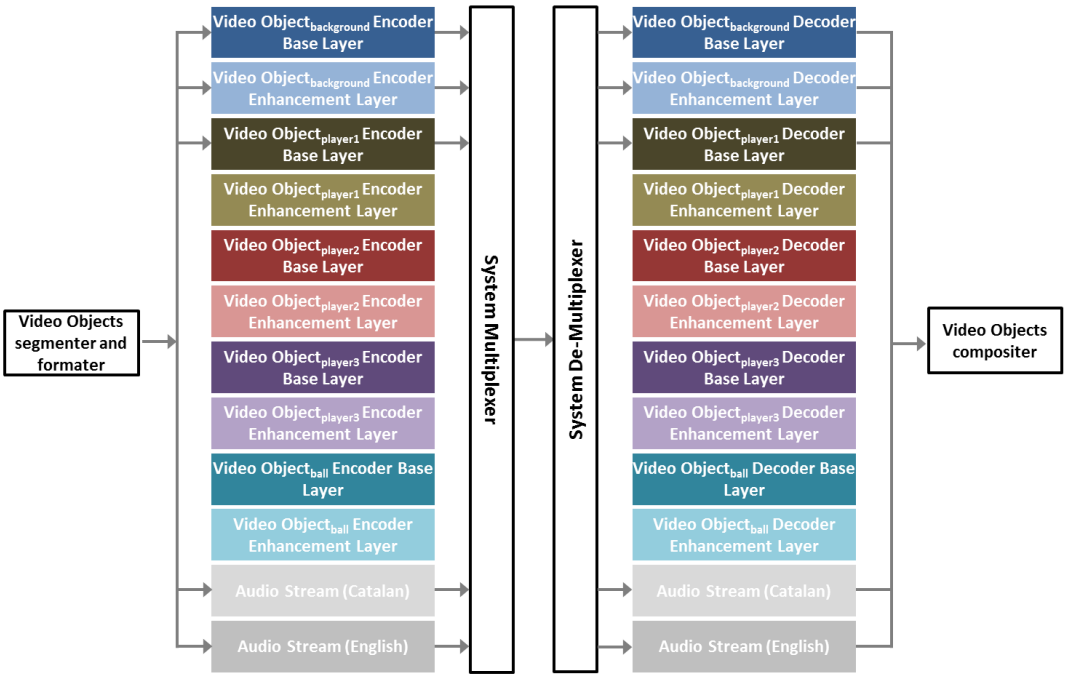


Figure 2.16: Block Diagram of VO encoders following the example in 2.14 based on Figure 2.14 in [34]

OCI descriptors can be conveyed in Object descriptors, Elementary Stream Descriptors or, if they are time variant, in the Elementary Streams. Multiple object descriptors and events can be bound up with the same OCI descriptor to constitute small and synchronised entities [33].

IPMP descriptor The purpose of IPMP is to provide intellectual property management and protection tools to the terminal. The IPMP system consists on IPMP elementary streams and descriptors conveyed as part of the Object Descriptor Stream [33]. It provides ES media standard identification information.

SL descriptor The SL Descriptor conveys configuration information for the Sync Layer. The information is key for ES synchronisation. It is described in more detail in Section 3.7.

Decoder descriptor This contains information about the media decoder for the related ES such as stream type and object type. It provides decoder-specific information to the media decoder for the linked media ES such as media type, MPEG-4 level and profile.

Examples of stream type include Object Descriptor Stream (0x01), Clock Reference Stream (0x02), Scene Description Stream (0x03), Visual Stream (0x04) or Audio Stream (0x05). Examples of object types include BIFS (0x01), visual ISO/IEC 14496-2 (0x20), ISO/IEC 14496-10 (0x21) or audio ISO/IEC 14496-3 (0x40). Note that different object type BIFS (0x01) always

```
class DecoderConfigDescriptor extends BaseDescriptor : bit(8)
      tag=DecoderConfigDescrTag {
    bit(8) objectTypeIndicator;
    bit(6) streamType;
    bit(1) upStream;
    const bit(1) reserved=1;
    bit(24) bufferSizeDB;
    bit(32) maxBitrate;
    bit(32) avgBitrate;
    DecoderSpecificInfo decSpecificInfo [0 .. 1];
    profileLevelIndicationIndexDescriptor profileLevelIndicationIndexDescr [0 .. 255];
}
```

Table 2.12: DecoderConfig Descriptor [33]

have stream type 0x03.

Fig. 2.15 shows the *DecoderConfig* Descriptor within the ES Descriptor. The example shows how an ES Descriptor with *streamType*=0x04 (Visual Stream) and *objectTypeIndicator*=0x21 (ISO/IEC 14496-10) conveys the related *AVCDecoderSpecificInfo* (with AVC decoder information). The *DecoderConfig* Descriptor is found in Table 2.12.

There are multiple decoder config descriptors such as AVCDecoderSpecificInfo (AVC streams), BIFSConfigEx (for BIFS streams), or AFXConfig (for Animation Framework Extension streams).

QoS descriptor Establishes the QoS requirements for the related ES. The parameters are: maximum and preferred end-to-end delay (ms), allowed AU probability loss, maximum and average AU size, maximum AUs arrival rate (AUs/s) as well as the ratio to fill the buffer in case of pre or re-buffering.

Extension descriptor A generic descriptor used for specific applications and future use.

2.3.2.4 T-STD

Transport System Target Decoder (T-STD) for delivery of ISO/IEC 14496 program elements encapsulated in MP2T streams is further explained in MPEG-2 part 1 'Systems'. The T-STD is visualised in Fig. 2.17 and Table 2.13 describes the variable names.

Processing of FlexMux Streams As described in Fig. 2.17, the Transport Stream demultiplexer delivers the FlexMux Stream n to its transport buffer TB_n, following this, the FlexMux Stream is delivered to the MB_n buffer at a rate RX_n, established at *TB_leak_field_rate* in the MultiplexerBuffer Descriptor. In this buffer, PES packets or 14496_sections packets are delivered, however any duplicate TS packets are discarded. The size of buffer differs: TB_n has a fixed size of 512 bytes whereas MB_n has a variable value defined in *MB_buffer_size* in the

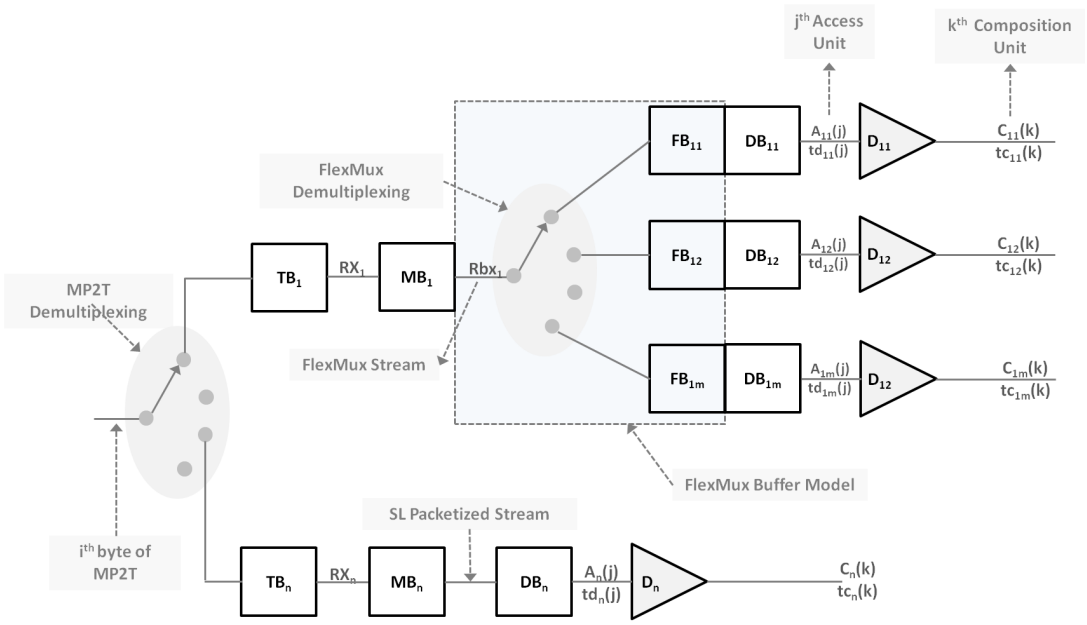


Figure 2.17: Transport System Target Decoder (T-STD) for delivery of ISO/IEC 14496 program elements encapsulated in MP2T. Figure 1 in [30]. The variables in T-STD are described in Table 2.13

MultiplexerBuffer Descriptor.

Data from MB_n are delivered to their correspondent FB_{pn} buffer at Rbx_p bit rate. Rbx_p is indicated in field *fmxRate* in each FlexMux Stream following the FlexMux Buffer Model and shall apply to all packets from the same FlexMux stream. Data leaves the FlexMux buffer model and enters in the decoding buffer, DB_{pm}, of each correspondent stream, subsequently decoding will be performed at indicated Decoding Timestamp (DTS) time, transforming access units (AU) into composition units (CU) and finally, the CUs ready to go though the composition process at the corresponding Composition Timestamp (CTS) time [30].

Processing of SL-Packetised Streams As shown in bottom half of Fig. 2.17, the Transport Stream demultiplexer delivers the SL-packetised Stream n to its transport buffer TB_n; following this, the SL-packetised Stream is delivered in a similar manner to above.

In the case of SL-packetised streams the data flows from MB_n buffer to the decoding buffer, DB_n, where it will leave at DTS time to be decoded and finally sent to the composition process at the corresponding CTS time.

Carriage within a Transport Stream Multiple programs, specified at the Program Map Table (PMT), can be carried within a MP2T stream. TS can convey among the already defined streams, 14496 content. 14496 content can be conveyed by different programs within one MP2T

Variable	Meaning
$\mathbf{TB_n}$	'transport buffer'
$\mathbf{MB_n}$	'the multiplex buffer for FlexMux stream n or for SL-packetized stream n'
$\mathbf{FB_{np}}$	'the FlexMux for the ES in FlexMux channel p of FlexMux stream n'
$\mathbf{DB_{np}}$	'the decoder buffer for the elementary stream in FlexMux channel p of FlexMux stream n'
$\mathbf{DB_n}$	'the decoder buffer for elementary stream n'
$\mathbf{D_{np}}$	'the decoder for the elementary stream in FlexMux channel p of FlexMux stream n'
$\mathbf{D_n}$	'the decoder for elementary stream n'
$\mathbf{Rx_n}$	'the rate at which data are removed from TB_n'
$\mathbf{Rbx_n}$	'the rate at which data are removed from MB_n'
$\mathbf{A_{np}(j)}$	'the j^{th} access unit in elementary stream in FlexMux channel p of FlexMux stream n. $An_p(j)$ is indexed in decoding order'
$\mathbf{A_n(j)}$	'the j^{th} access unit in elementary stream n. $A_n(j)$ is indexed in decoding order'
$\mathbf{Td_{np}(j)}$	'the decoding time, measured in seconds, in the system target decoder of the j^{th} access unit in elementary stream in FlexMux channel p of FlexMux stream n'
$\mathbf{Td_n(j)}$	'the decoding time, measured in seconds, in the system target decoder of the j^{th} access unit in elementary stream n'
$\mathbf{C_{np}(k)}$	'the k^{th} composition unit in elementary stream in FlexMux channel p of FlexMux stream n. $C_{np}(k)$ results from decoding $An_p(j)$. $C_{np}(k)$ is indexed in composition order'
$\mathbf{C_n(k)}$	'the k^{th} composition unit in elementary stream n. $C_n(k)$ results from decoding $A_n(j)$. $C_n(k)$ is indexed in composition order'
$\mathbf{tc_{np}(k)}$	'the composition time, measured in seconds, in the system target decoder of the k^{th} composition unit in elementary stream in FlexMux channel p of FlexMux stream n'
$\mathbf{tc_n(k)}$	'the composition time, measured in seconds, in the system target decoder of the k^{th} composition unit in elementary stream n'
$\mathbf{t(i)}$	'the time in seconds at which the i^{th} byte of the Transport Stream enters the system target decoder'

Table 2.13: Notation of variables in the MPEG-4 T-STD [30] for Fig. 2.17

stream having as each program has a unique PID [30].

A 14496-1 scene is specified by an Initial Object Descriptor moreover the content of a 14496 program is indicated by the program's PMT within the MP2Ts. The 14496 content is identified by *stream_type* at the PMT plus the PID value. *Stream_type=0x12* relates to PES within the MP2Ts containing SL or FlexMux stream whereas *stream_type=0x13* describes 14496_sections within the M2TS containing object description stream or scene description stream, as indicated in Table 2.14.

Two types of data are conveyed within 14496_sections, an Object Descriptor Stream and Scene Descriptor Stream. The field *table_id* in the section header signifies the type. A 14496_section can only convey one SL-packet or multiple FlexMux packets. The presence of a SL or Flex-

14496 Stream	Packetisation		Stream Type	Stream/Table Id
Object Descriptor	SL	PES	0x12	stream_id=111 1010
		14496_sections	0x13	table_id=0x05
	FlexMux	PES packets	0x12	stream_id=111 1011
		14496_sections	0x13	table_id=0x05
Scene Descriptor	SL	PES packets	0x12	stream_id=111 1010
		14496_sections	0x13	table_id=0x04
	FlexMux	PES packets	0x12	stream_id=111 1011
		14496_sections	0x13	table_id=0x04
Other Stream	SL	PES packets	0x12	stream_id=111 1010
	FlexMux	PES packets	0x12	stream_id=111 1011

Table 2.14: ISO/IEC defined options for carriage of an ISO/IEC 14496 scene and associated streams in ITU-T Rec. H.222.0. ISO/IEC 13818-1 from Table 2-65 in [30]

Mux Channel (FMC) descriptor indicates the type of payload and additionally for every 14496 stream, it identifies the ES_ID. A list summarising the carriage of MPEG-4 streams (objects, scene and other, including media) within MP2T stream is found in Table 2.14.

Content access procedure for 14496 program components within MP2Ts There are a logical sequence of functions to be undertaken when a 14496 program is received [30]. These are:

- Obtain the program's PMT
- Determine the Initial Object Descriptor (IOD) of the initial descriptor loop
- Establish the object descriptor's ES_IDs, scene description and streams specified within the first object descriptor
- Obtain, from all elementary_PIDs, all SL descriptors and FlexMux Channel (FMC) descriptors from the second descriptor loop
- Generate a stream map table from descriptors between ES_IDs and related elementary_PID and FlexMux, if needed
- Employ ES_ID to place the Object Descriptor Stream and its Stream Map Table.
- Find, using the ES_ID and stream map table, all streams described in the Initial Object Descriptor
- Identify ES_IDs of additional streams through the Object Descriptor Stream
- Find supplementary streams by their ES_ID and the stream map table

```
aligned(8) class Box (unsigned int(32) boxtype, optional unsigned int(8)[16]
                extended_type) {
    unsigned int(32) size;
    unsigned int(32) type = boxtype;
    if (size==1) {
        unsigned int(64) largesize;
    } else if (size==0) {
        // box extends to end of file
    }
    if (boxtype==‘uuid’) {
        unsigned int(8) usertype = extended_type;
    }
}
```

```
aligned(8) class FullBox (unsigned int(32) boxtype, unsigned int(8) v, bit(24) f)
                extends Box(boxtype) {
    unsigned int(8) version = v;
    bit(24) flags = f;
}
```

Table 2.15: Box and FullBox class [12]

2.3.3 MPEG-4 part 12: ISO Base Media File Format

The ISO Base File format is defined as ‘*a base format for media file formats*’, that ‘*contains the timing, structure, and media information for timed sequences of media data, such as audio-visual presentation*’ [12]. This file format aims to be independent from network protocols.

Within the ISO Base File Standard *brands* are also defined. A brand is a group of requirements within the ISO Base file system. A file conforms to a brand if all the brand’s requirements are met. Each brand supports a ISO subset of structural boxes. Only a finite number of brands are defined in ISO Standard, other media specifications may define different ISO brands used.

The ISO file format is made of objects which are called *boxes*. All data within the file is within a box. There are multiple boxes and following a specific hierarchy, though here, only the most relevant ones and the time related ones are explained. The specification of the boxes use the Syntax Description Language (SDL), also defined in MPEG-4 [12].

Every box or object, contains a header which provides the *size* and *type* fields. The data types used in the boxes allow the compact type (32-bit size) or the extended type (64-bit size). Usually only the Media Data Box requires the extended data type. The structure of a box is found in Table 2.15.

Together with the Box class, the *FullBox* provides extra version and flag values when needed. The version is set to zero when 32-bit fields are used in the box and one for 64-bit fields. The structure of the *FullBox* is found in Table 2.15.

One of the obligatory boxes in any ISO File is the File Type Box (*ftyp*). Only one such box per file is required and it should be located at the beginning of the file. The structure of *ftyp*

```
aligned(8) class FileTypeBox extends Box ('ftyp') {
    unsigned int(32) major_brand;
    unsigned int(32) minor_version;
    unsigned int(32) compatible_brands[ ]; // to end of the box
}
aligned(8) class MediaDataBox extends Box( mdat) {
    bit(8) data[ ];
}
```

Table 2.16: Box and FullBox class

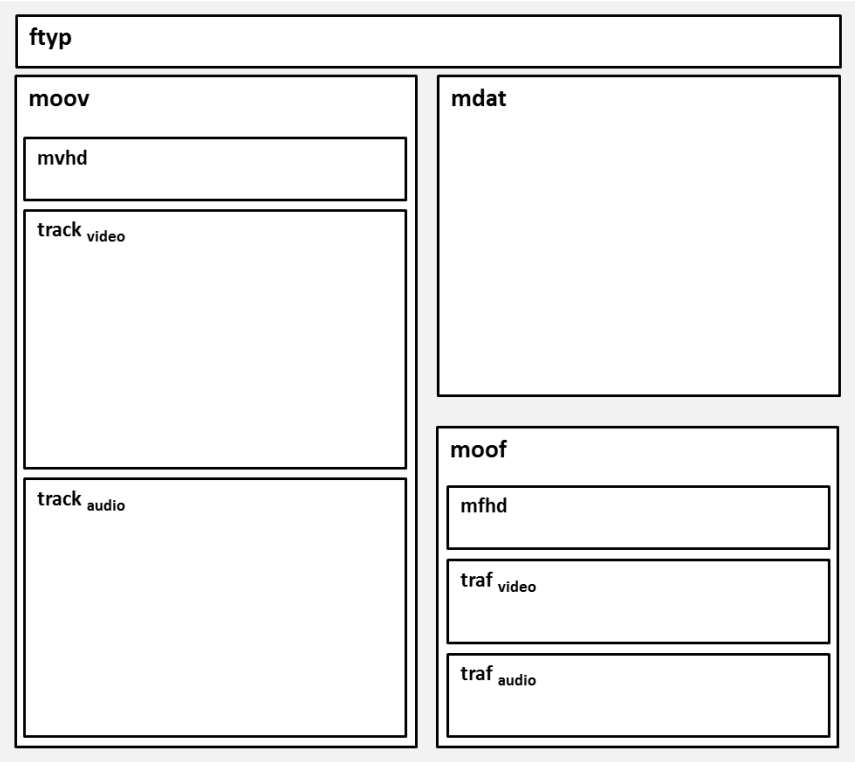


Figure 2.18: ISO File Structure example

is found in Table 2.16 [12].

The *major_brand* indicates the brand identifier and the *minor_version* the version of the brand, whereas the *compatible_brands* is a list of brands compatible with the ISO file.

The box that contains the media data is the Media Data Box (*mdat*). There can be zero or multiple *mdat* boxes within a presentation. The structure of *mdat* is found in Table 2.16 [12].

In Fig. 2.18, an example of a high level structure of an ISO Base File format is shown. The structure is all based on boxes. As mentioned, the *ftyp* box is always compulsory and placed at beginning of the file. In Fig. 2.19, the file structure used by MS-SSTR Adaptive Streaming

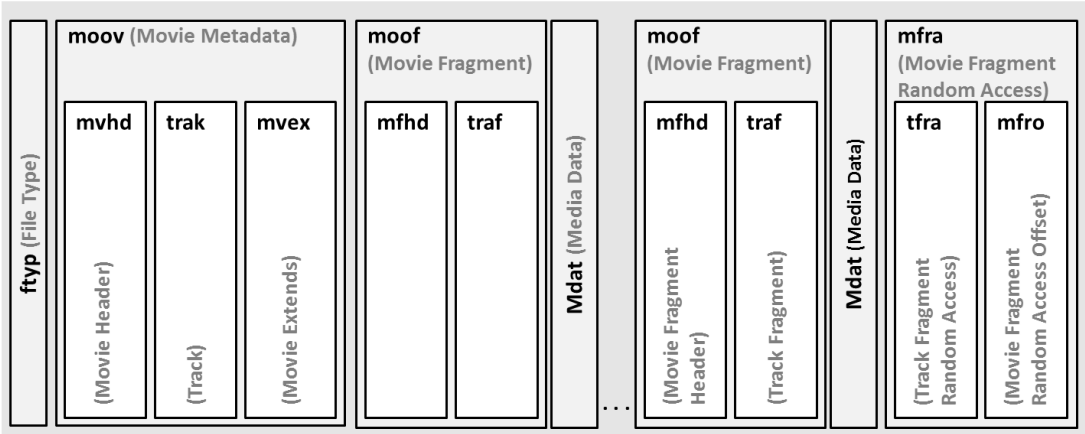


Figure 2.19: ISO File system used by MS-SSTR [35]

protocol is shown, which uses the ISO file format for the media delivery.

In Fig. 2.20 the information extracted from an MP4 file following the ISO file format can be seen. The video analysed is 52.209s long. On the left, the overall ISO file structure of the example can be seen (a brief description is included). On the right of the figure information (some fields values) from relevant boxes is included.

The boxes *ftyp*, *free* and *mdat* are boxes related to the entire media file. The *mdat* box contains the media samples and finally, *moov* box (meta-data container) contains other boxes such as the *mvhd*, two *tracks* and *udta* (user-data information).

In the *ftyp* box the ISO brand and the compatible brands are listed. The box *mdat* contains the media samples of the two tracks (media streams). The *stbl1* (video) contains 1253 samples and the *stbl2* (audio) contains 2435 samples.

Track1 contains the information about an AVC visual stream whereas *track2* contains the AAC audio stream information. The AVC video information is located in box *avc1* (AVC visual sample entry) whereas AAC audio information is located in *esds* (AAC audio decoder initialization information).

The boxes *mvhd*, *tkhd*, contain time information and *stts* and *ctts* contain timestamps. In Chapter 3, Section 3.8 the boxes within the example will be further explained.

2.3.4 MP3 Audio File Format

In this section, the MP3 audio format is described. MPEG Audio Layer 3, commonly known as MP3 is one of the most used audio formats in Internet Radio and the one used in the prototype.

The MP3 frame has a 4-byte MP3 header that provides information about the audio characteristics. In Fig. 2.21, the structure of the MP3 header with all header fields is shown.

The MP3 header has the following fields: *SyncWord* (11-bit), *MPEG version* (2-bit), *Layer* (2-bit), *protection_bit* (1-bit), *Bitrate_index* (4-bit), *sampling_frequency* (2-bit), *padding_bit* (1-

Figure 2.20: ISO File example structure and box content

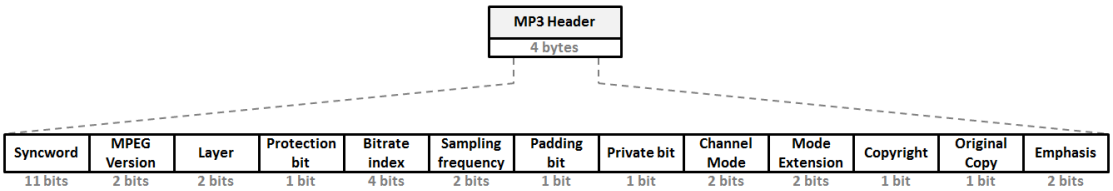


Figure 2.21: MP3 Header structure

bit), *private_bit* (1-bit), *channel mode* (2-bit), *mode_extension* (2-bit), *copyright* (1-bit), *original/copy* (1-bit), and *emphasis* (2-bit).

The *SyncWord* is all set to one. The *MPEG version* possible values are:

- 00 → Unofficial version of MPEG 2.5
- 01 → reserved
- 10 → MPEG version 2 (13818-3)
- 11 → MPEG version 1 (11172-3)

The channel mode field is based the following:

- 00 → Stereo
- 01 → Joint stereo (Stereo)
- 10 → Dual channel (Stereo)
- 11 → Single channel (Mono)

The Samples per Frame (SpF) is given by version and layer as shown in Table 2.17. The MP3 frame size in bytes can be derived from the SpF or the bitrate along with the sample rate plus the value of the padding as described in the following examples.

$$MP3_{frameSize} = (\frac{(SpF/8) \cdot bitRate}{SamplingFrequency} + padding) \tag{2.1}$$

When audio is MP3 Layer I, the equation for the frame size is:

$$MP3_{frameSize} = \frac{12 \cdot bitRate}{SamplingFrequency} + padding \tag{2.2}$$

When audio is MP3 Layer II and III, the equation for the frame size is:

$$MP3_{frameSize} = \frac{144 \cdot bitRate}{SamplingFrequency} + padding \tag{2.3}$$

$$MP3_{frameLength}(ms) = \frac{SpF}{SamplingFrequency(Hz)} \cdot 1000 \tag{2.4}$$

	Layer 1	Layer 2	Layer 3
MPEG-1	384	1152	1152
MPEG-2	384	1152	576

Table 2.17: MP3 Samples per Frame (SpF)

Bits	MPEG-1	MPEG-2	MPEG-2.5
00	44100	22050	11025
01	48000	24000	12000
10	32000	16000	8000
11	Reserved	Reserved	Reserved

Table 2.18: MP3 Sampling Rate Frequency (Hz)

	MPEG-1			MPEG-2, 2.5	
Bits	**Layer I**	**Layer II**	**Layer III**	**Layer I**	**Layer II-III**
0000	Free	Free	Free	Free	Free
0001	32	32	32	32	8
0010	64	48	40	48	16
0011	96	56	48	56	24
0100	128	64	56	64	32
0101	160	80	64	80	40
0110	192	96	80	96	48
0111	224	112	96	112	56
1000	256	128	112	128	64
1001	288	160	128	144	80
1010	320	192	160	160	96
1011	352	224	192	176	112
1100	384	256	224	192	128
1101	416	320	256	224	144
1110	448	384	320	256	160
1111	Reserved	Reserved	Reserved	Reserved	Reserved

Table 2.19: MP3 Bit Rate (kbps) Table

The values for SpF can be found in Table 2.17, Sampling Frequency in Table 2.18 and, finally, in Table 2.19 the values for MP3 Bitrate are enumerated.

As an example, the values from the MP3 file used in the proof-of-concept prototype are: SampleRate=44.1k BitRate 128k, SamplePerFrame=1152.

$$MP3_{frameLength}(ms) = \frac{SpF}{Sample(Hz)} \cdot 1000 = \frac{1152}{44100} \cdot 1000 = 26.12ms \tag{2.5}$$

$$MP3_{frameLength}(bytes) = \frac{144 \cdot 128000}{44100} + padding = 417bytes + padding \tag{2.6}$$

PID	MP2T Packets	%	Content
000 (0x00)	1349	0.42	PAT
100 (0x64)	1349	0.42	PMT
101 (0x65)	292013	90.69	MPEG-2 Video
102 (0x66)	23603	7.33	MPEG-1 Audio (English)
103 (0x69)	3668	1.14	MPEG-1 Audio (Visual impaired commentaries)

Table 2.20: Analysis Real Sample MP2T stream duration 134s (57.7M)

2.3.5 DVB-SI and MPEG-2 PSI

DVB, independently of the delivery platform used, —Terrestrial, Satellite, Cable or IPTV, performs the media delivery via MPEG-2 Systems (MP2T streams). The only difference between DVB (Satellite, Terrestrial, and Cable) and DVB-IPTV is the recommended use of RTP protocol as a transport protocol for IPTV [36] [37] [38] [39].

The details of the audio, video codec system used within a program is transmitted via DVB Service Information (SI) and MPEG-2 Program Specific Information (PSI) tables. The relationship between both tables structures is shown in Fig. 2.22 and the distribution of DVB-SI and MPEG-2 PSI tables in a MP2T stream is shown in Fig. 2.23. As a real example analysis in Table 2.20 it is found the number of DVB-SI and MPEG-2 PSI packets in a MP2T stream.

To modify any media at client-side (as it is done in the prototype) the first step to be performed is to modify the DVB Service Information (DVB-SI) [40] and MPEG-2 Program-Specific Information (MPEG-2 PSI) tables [30]. As an example, in the prototype the PMT table is modified to reflect the addition of a new audio stream. Those tables convey the fundamental information for the decoder to perform the play-out of any media received at client-side. More details are provided in next sections.

2.3.5.1 DVB-SI

DVB-SI tables include some obligatory and optional tables. Table 2.21 describes all SI tables; DVB Storage Media Inter-operability (DVB SMI) tables are also included although not used in the prototype. All table definitions are taken from [40].

Appendix B lists information for Table SDT in Table 4, Table EIT in 5, Table TDT in 6 and Table TOT in 7.

2.3.5.2 MPEG-2 PSI

MPEG-2 PSI tables also include some obligatory and optional tables as shown in Table 2.22. All tables are transmitted within MP2T packets within the video stream and each MP2T only conveys one table. The structure of the Table PMT is found in Table 8 and the Table PAT in Table 9 in Appendix B.

For the prototype developed in this research only the PMT table needs to be modified at

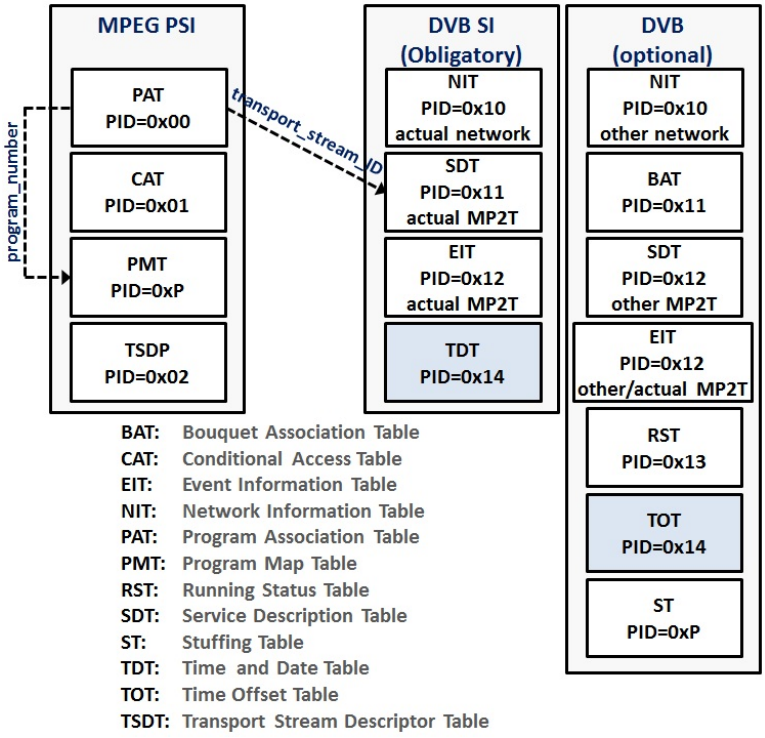


Figure 2.22: DVB-SI and MPEG-2 PSI relationship tables [40]

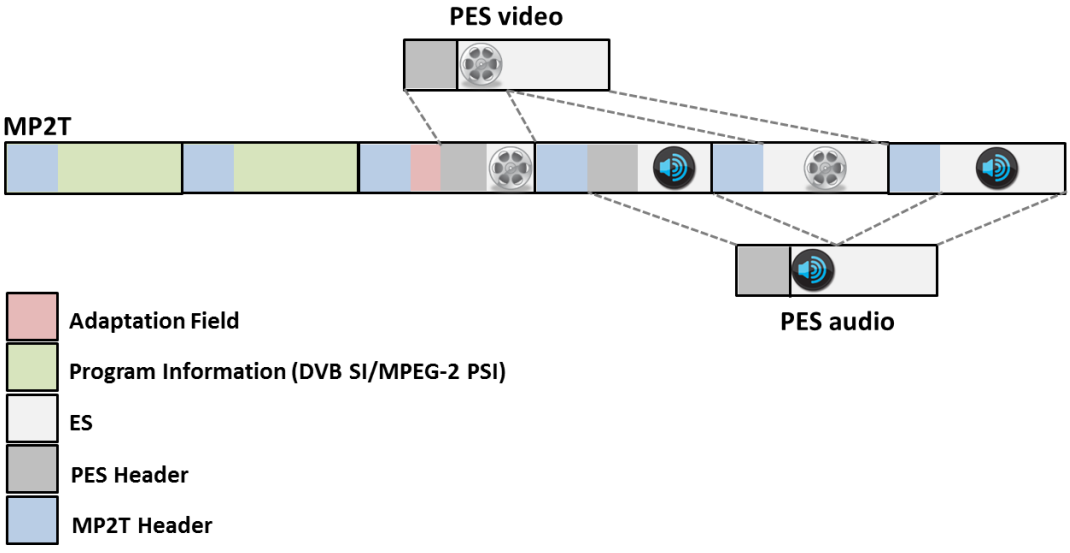


Figure 2.23: DVB-SI and MPEG-2 PSI distribution in a MP2T stream

client-side adding the required components, i.e., extra audio streams. The SDT and PAT tables, although being streamed, don't require modification by the prototype because no extra service

	Table		Description	DVB
Obligatory	NIT	Network Information Table	It details Network information and about the multiplexed TS streamed over the Network	DVB-SI
	SDT	Service Description Table	It specifies information about the services conveyed within the TS or other TSs	DVB-SI
	EIT	Event Information Table	It conveys the information about the events chronological schedule	DVB-SI
	TDT	Time Description Table	It provides UTC-time and date information	DVB-SI
Optional	BAT	Bouquet Association Table	It describes a group of services called bouquet	DVB-SI
	TOT	Time Offset Table	It provides UTC-time information and local time offset	DVB-SI
	RST	Running Status Table	To precise and fast actualization of time events status	DVB-SI
	ST	Stuffing Table	To cancel present sections	DVB-SI
	DIT	Discontinuity Information Table	to signal transitions points in discontinuous SI information	DVB SMI
	SIT	Selection Information Table	It details services and event of partial TSs	DVB SMI

Table 2.21: DVB-SI Tables [40]

	Table		Description
Obligatory	PAT	Program Association Table	It creates the link between Program Number and the Program Map Table
	PMT	Program Map Table	Indicates PID values for program components
Optional	TSDT	Transport Stream Descriptor Table	
	NIT	Network Information Table	IT conveys Physical Network Information
	IPMP	Control Information Table	Conveys IPMP tool list, rights container
	CAT	Conditional Access Table	Links encrypted conditional access information with PID values via Entitlement Management Message (EMM) streams

Table 2.22: MPEG-2 PSI Tables [30]

or program is added to the MP2T stream received. More details about MPEG-2 PSI table in the prototype will be explained in Chapter 4.

Of particular relevance in the PMT Table is the field *PCR_PID* (13 bits). Every MP2T has associated with it the PID of one program, all PCRs will be conveyed within MP2T packets of this PID program.

The SDT Table advertises all services within a MP2T stream. It could include services[1] from the actual or other MP2T. One service can include multiple programs[2].

The EIT Table advertises all program events within a MP2T stream. It could include events from the actual or other MP2T. There are two types of events, *present/following* and *event schedule information*. The *present/following* table lists the information about the present and following event within the Service. Similarly, the *event schedule information* contains the event schedule for present and following events. The field *duration* (24-bit) represents the time in hours (first byte), minutes (second byte) and seconds (third byte), e.g., *06:08:10* duration will be *0x060810*.

2.3.5.3 DVB-SI Time related Tables

The two time related tables in DVB-SI are Time to Date Table (TDT) and Time Offset Table (TOT). The first provides the time of transmission and the later provides the time offset of the area receiving the DVB stream. The structure of TDT is found in Table 6 and TOT structure in Table 7 in Appendix B.

The TDT has a *UTC_time* (40 bits) field, which conveys the UTC time of the DVB transmission. The TOT also includes the fields *UTC_time* but includes the descriptor *Local Time Offset* Descriptor which provides the country information *country_code* and *country_region_id* and the local time offset, the offset via *local_time_offset* and the *local_time_offset_polarity*.

The UTC field use the UTC and Modified Julian Date (MJD) format. '*This field is coded as 16 bits giving the 16 LSB of MJD followed by 24 bits coded as 6 bits coded as 6 digits in 4 bit Binary Coded Decimal (BCD)*'. It important to note that the granularity of UTC values used in TOT and TOT tables is seconds.

The MJD is a variation of the Julian Date (JD). The JD counts the number of days since the Julian Date (noon at 1st January 4713 BC). The MJD has a few modifications, first it begins at midnight and removes the first two digits. Therefore, the formula to transform JD to MJD is the following:

$$MJD = JD - 2400000.5 \tag{2.7}$$

For example, the 31st July of 1976 is 42990 in MJD format.

The frequency with which the tables are inserted in the DVB/MPEG-2 stream have different restrictions based on the table type. The requirements for each table are listed in Table 2.23, 25ms is the minimum interval for all the tables. Moreover, the maximum interval varies from 0.5s for PAT, PMT and CAT to 30s for TDT and TOT tables.

[1] '*Sequence of programmes under the control of a broadcaster which can be broadcast as part of a schedule*' [40]

[2] '*Concatenation of one or more events under the control of a broadcaster e.g., news show, entertainment show*' [40]

	Table	Maximum interval	Minimum interval
MPEG-2 PSI	PAT	0.5s	25ms
	PMT	0.5s	25ms
	CAT	0.5s	25ms
DVB-SI	NIT	10s	25ms
	BAT	10s	25ms
	SDT actual multiplex	2s	25ms
	SDT other MP2T	10s	25ms
	EIT present/following table	2s	25ms
	EIT schedule table	10s	25ms
	RST	-	25ms
	TDT	30s	25ms
	TOT	30s	25ms
	ST	-	-

Table 2.23: Timing DVB-SI and MPEG-2 PSI Tables [30] [40] [41]

2.3.6 MMT

MPEG Media Transport (MMT) aims to provide a unique solution for multimedia content over heterogeneous networks, both broadcast and broadband delivery platforms. MMT is the MPEG Standard 23008 part 1, recently approved in 2014 [42].

There are four layers within the MMT architecture. The Media Coding Layer (C-Layer), Delivery (D-Layer), Encapsulation (E-Layer) and the Signalling Layer (S-Layer).

In the E-Layer, where ISO Base Media File Format (ISO BMFF) is used, the content's logical structure and the physical encapsulation format is specified in [43].

Within the D-Layer, the application layer protocol provides streaming delivery of packetised media content [43]. The encapsulation functions establish the boundaries for fragmentation for its structure agnostic packetisation [44]. Within the D-Layer there are three sub-layers:

- D1: Generates the MMT payload
- D2: QoS and Timestamp delivery. Generates the MMT Transport Packet
- D3: Support cross-layer optimization exchanging QoS-related information between the application layer and the network layers

The S-Layer is the cross-layer interface between D-Layer and E-Layer. S-Layer is structured in S1 and S2. S1 manages presentation sessions and S2 handles delivery sessions exchanged between end-points [45]. In Fig. 2.24 the structure is drawn. The time related fields have been included next to the related MMT Layer.

An MPU contains one or multiple MFU, moreover, a MFU can contain one of multiple AUs. A MPU always contains a number of complete AUs (See Fig. 2.25).

The MMT Logical Structure contains the following elements: Asset Delivery Characteristics

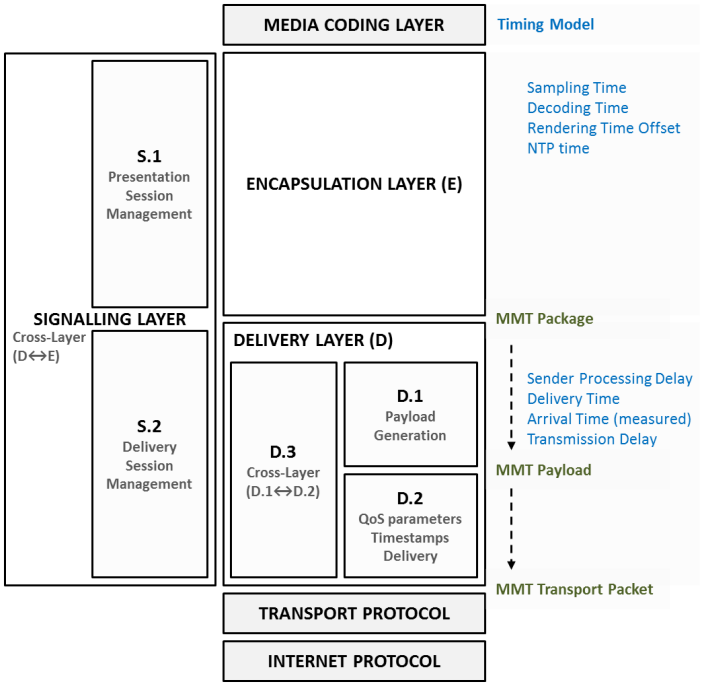


Figure 2.24: MMT Architecture from [44]

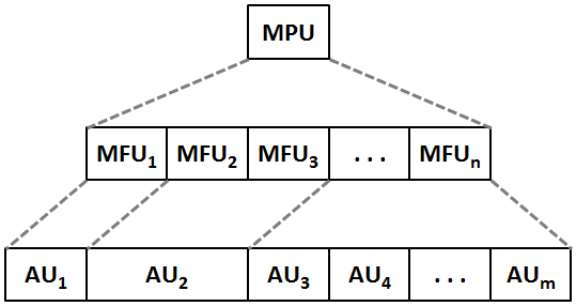


Figure 2.25: Relationship between MPU, MFU and media AUs

(ADC), MMT assets, Composition Information (CI), Media Fragment Unit (MFU) and Media Processing Unit (MPU). The complete MMT Logical Structure can be found in Fig. 2.26.

The MMT Packet represents the logical structure of the MMT Asset. Within the MMT packet there are the MMT Assets along with the CI and ADC, all linked to the MMT assets. The MMT asset provides the logical structure to convey the coded media data and also identifies multimedia data. MPU is the self contained data unit within the MMT asset. The D-Layer

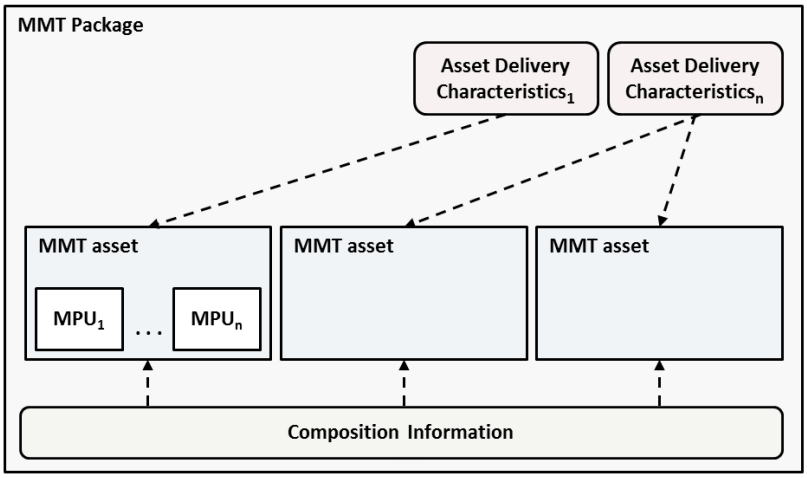


Figure 2.26: MMT Logical Structure of a MMT Package [45]

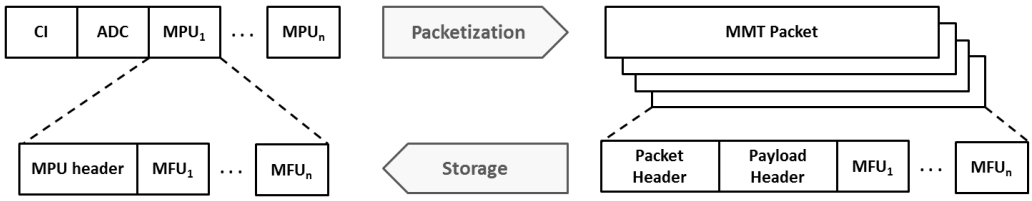


Figure 2.27: MMT Packetisation [45]

processing information of the MMT assets is provided by CI and ADF [44].

MMT storage system is a MMT file which contains all the MMT logical information such as the CI, ADC and related MPUs (composed of multiple MFUs). The MMT packetisation process generates MMT packets ready for real-time streaming, the information within the MMT file is packetised within a MMT packet adding a MMT packet header and a MMT payload header.

In Fig. 2.27 the process of packetisation to storage and vice versa is shown.

MMT also aims to unify broadcast and broadband media delivery by representing a common delivery tool for both media delivery systems. In Fig. 2.28, possible scenarios for broadcast MMT delivery are compared [46].

In Fig. 2.28 all possible options for packetisation in Broadcasting Systems are drawn. Although broadcast technologies are outside the scope of this thesis, MMT aims to find a common delivery with broadband techniques. On top of the Channel Coding and Modulation there are four choices. The first one packetises MMT directly over the Channel Coding, the second one conveys MMT packets over MP2T and over the Channel Coding. Finally, there is the option to use IP packetisation over MP2T packets or over Type Length Value (TLV) packets.

MMT also specifies a packet structure for media delivery. In Fig. 2.29 it is drawn the

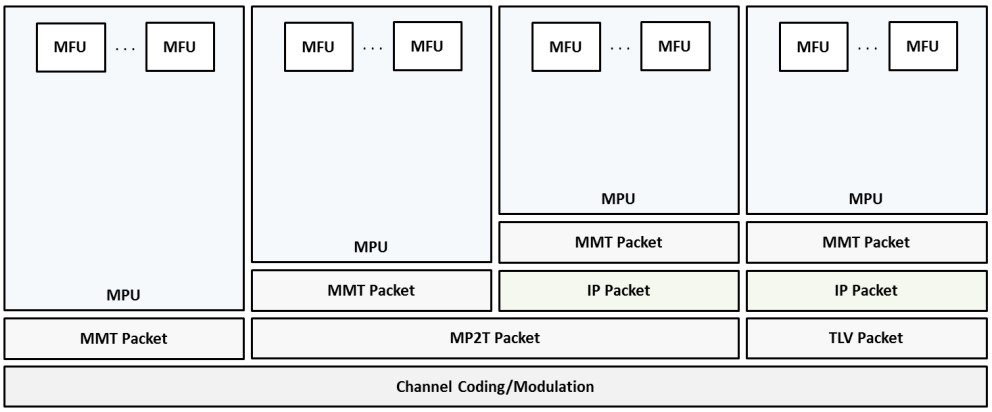


Figure 2.28: Comparison of Transmitting Mechanisms of MMT in Broadcasting Systems based on Table II from [46]

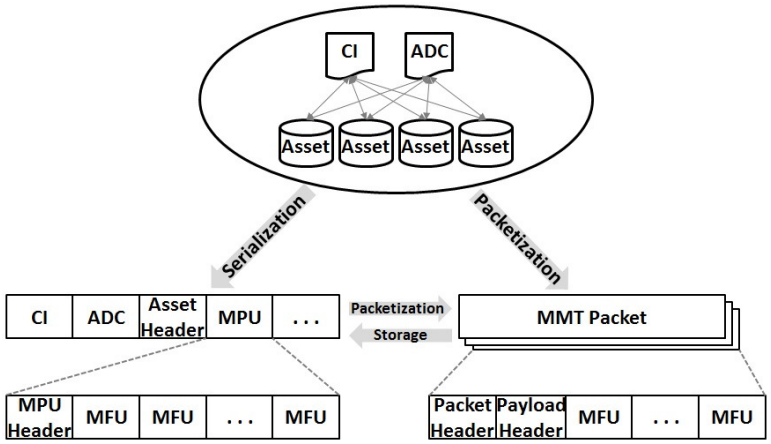


Figure 2.29: Relationship of an MMT package' storage and packetised delivery formats [43]

relationship between MMT storage package structure and the MMT package delivery format.

2.4 Transport Protocols

2.4.1 RTP (Real-Time Transport Protocol)

RFC 3550 [47] defines the Real-Time Protocol (RTP) and Real-Time Control Protocol (RTCP). Both protocols support the delivery, either unicast or multicast, of real-time data, such as multimedia, with some QoS support over IP networks.

RTP delivers the real-time data which is conveyed within its payload whereas RTCP provides control information about the transmission of the data.

The RTP header includes payload type, especially important in multimedia to inform the

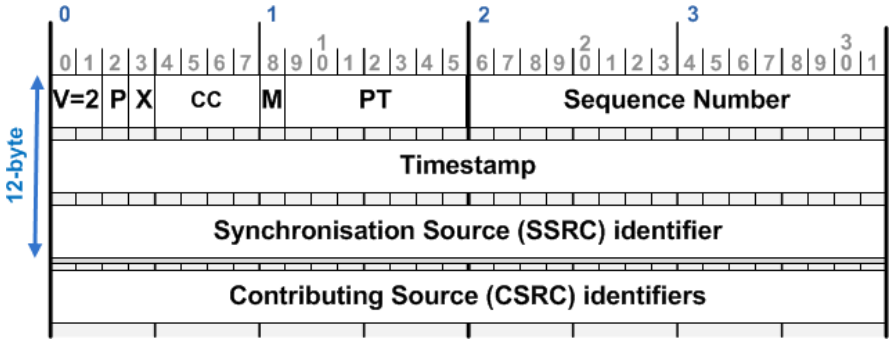


Figure 2.30: RTP Media packet [47]

receiver about payload content, sequence number, for packet loss and out-of-order monitoring, and timestamping for synchronisation purposes. Finally, RTP is typically carried over UDP for delay-sensitive, loss-tolerant traffic.

For the delivery of multimedia over IP Networks via RTP, it is essential for receivers to know the RTP payload content; consequently, there is the need to define the codes to assign a payload type to each payload format [47]. Every payload type specifies how to convey the media within RTP packets. E.g., the RTP payload for MP2T is 33 and the payload for MPEG Audio (MPA) is 14. This information is specified in different RFCs from the Internet Engineering Task Force (IETF) as shown in Section 2.4.3.

In Fig. 2.30 the RTP header fields are shown. In the context of this thesis, the most relevant fields are the *timestamp* (32-bit) and *payload* type (8-bit), the latter shown as PT [47].

2.4.1.1 RTP Timestamps

The timestamps is a 32-bit field coded within the RTP header. For security reasons the first value takes a random value.

Timestamp values, in the case of multimedia payload, specify the temporal relationship of content within the packet. In particular, they signify the sampling instant of the first media unit within the RTP payload.

Different multimedia streams will thus have independent timestamps with random initial offsets, therefore, synchronisation between multimedia streams from different sources cannot be accomplished with out further timing information.

2.4.2 RTCP (Real-Time Control Protocol)

In total, there are four types of RTCP packets, each with a specific function. These are receiver and sender *reports*, *description*, *application* and *goodbye* packets. The Report RTCP packets are used for various reasons; one primary reason is to enable RTP receivers to distribute reception quality feedback to other RTP senders. Another function relates to timing and lip-sync as

	RTCP Packet	PT
Report RTCP	SR (Sender)	200
	RR (Receiver)	201
Description RTCP	SDES	202
Good Bye RTCP	BYE	203
Application-Defined RTCP	APP	204

Table 2.24: RTCP Packet Types

Name	Description	Identifier	
CNAME	Canonical End-point Identifier SDES Item	CNAME=1	
NAME	User name SDES Item	NAME=2	
EMAIL	Electronic Mail Address SDES Item	EMAIL=3	
PHONE	Phone Number SDES Item	PHONE=4	
LOC	Geographic User Location SDES Item	LOC=5	
TOOL	Application/Tool Name SDES Item	TOOL=6	Name and version applications generating the stream
NOTE	Notice/Status SDES Item	NOTE=7	Informs source's state
PRIV	Private Extensions SDES Item	PRIV=8	To define application-specific SDES extensions

Table 2.25: SDES Packet Items, Identifier and Description [47]

described later. A list of RTCP report types is found in Table 2.24.

SR and RR packets have a common structure with some differences. SR packets have three sections, header, sender information and zero, one or multiple Report blocks, whereas, a RR packet shares the same structure with no Sender report information. Of particular importance to the proof-of-concept NTP and RTP timestamps are sent by SR within the sender's information section. RTCP SR structure is found in Fig. 2.31 and RTCP RR in Fig. 2.32.

The Source Description RTCP Packet (RTCP SDES) aims to transmit information describing the source. It has two sections, the header (32-bit) and zero or multiple chunks. The type of source information conveyed within an RTCP SDES is shown in Table 2.25.

The Application-Defined RTCP Packet (APP) is designed for testing purposes for new applications or features. Finally, the Goodbye RTCP packet (BYE) communicates to all receivers regarding the inactivity of a source [47].

RTP timestamps are numeric values used to recreate the intra-stream relationship of a media stream at the destination. As such, it is essentially a counter with no explicit link to a timescale. However, RTCP packets, which accompany the RTP packets with control information, provide this function in that they relate RTP timestamps to a wall-clock time via the RTP timestamp field and the two NTP fields, *NTP timestamp most significant* word and *NTP timestamp less significant word* with 32 bits each. See Fig. 2.31. Using RTCP RTP and the NTP timestamps,

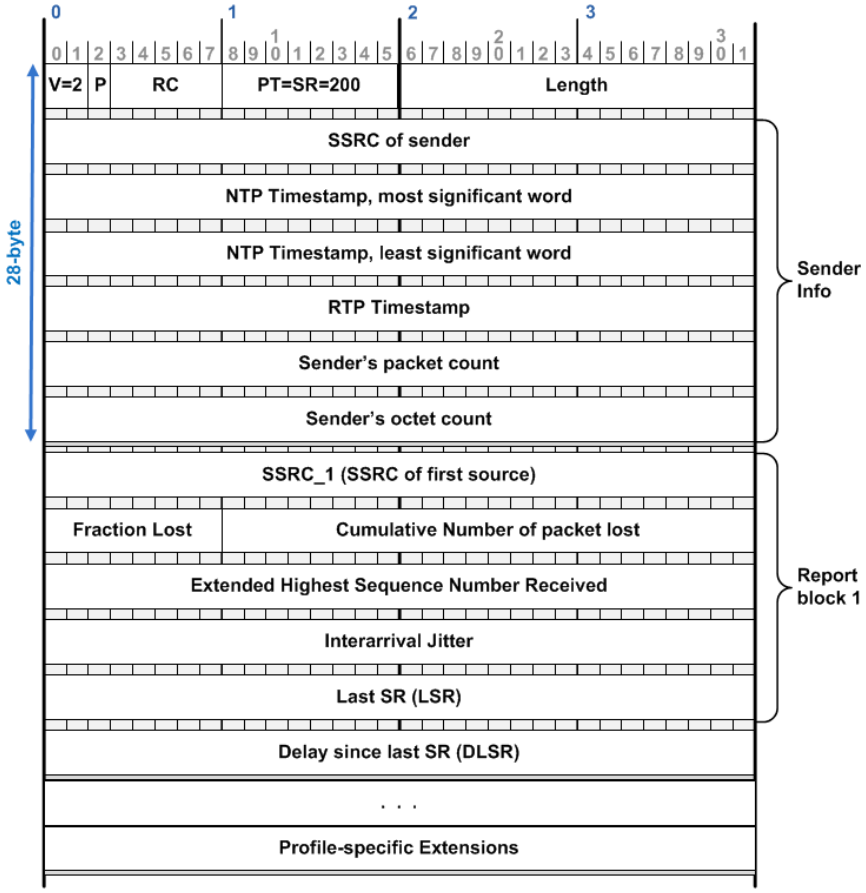


Figure 2.31: RTCP Sender Report packet [47]

therefore, the receiver has a mapping between the wall-clock time on the sender and the RTP timestamp. This feature is heavily used in the prototype for synchronisation and clock skew detection.

Two different Report RTCP structures can be found, Sender Report (RTCP SR) and Receiver Report (RTCP RR) depending on whether the sender of the RTCP packet is also a sender (former case) or not (latter case). See Fig. 2.31 and 2.32 for details.

There are two further timestamp fields in the RTCP SR report packets, *Last SR timestamp* (32-bit) and *Delay since last SR* (32-bit). The former encodes the 32 middle bits of the NTP wall-clock timestamp extracted from the most recent RTCP SR packet whereas the latter is the delay between the arrival of that $SSRC_n$ SR packet and the sending of reception report block for $SSRC_n$ [47]. The prototype does not utilise these timestamps.

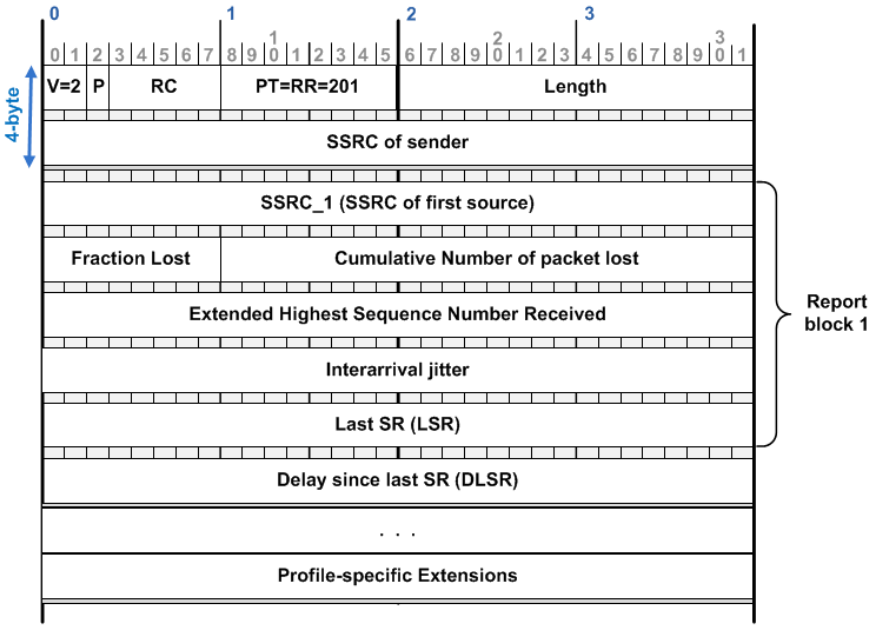


Figure 2.32: RTCP Receiver Report packet [47]

2.4.2.1 RTCP Packets Fields Related to QoS

As mentioned there are fields within the RTCP report block conveying useful information to monitor the QoS of the transmission. These are the fraction lost, cumulative number of packet lost and the inter-arrival jitter.

The fraction lost (8-bit) is the quantity of packets lost divided by the number of packets expected since last report packet was sent, the cumulative number of packets lost (24-bit) is the sum of packets lost since session began. Finally, the inter-arrival jitter (32-bit) is an unsigned integer of the variance of inter-arrival time of RTP packets calculated in timestamp units.

$$D(i,j) = (R_j - R_i) - (S_j - S_j) = (R_j - S_j) - (R_i - S_i) \tag{2.8}$$

$$J(i) = J(i-1) + \frac{(|D(i-1,i)| - J(i-1)}{16} \tag{2.9}$$

Where D is the *'difference in packet spacing at the receiver compared to the sender for a pair of packets'* [47], where S_i is the i_{th} sent packet, R_i is arrival time of i^{th} packet and, finally, *J(i)* is the jitter of the i^{th} packet[1].

[1]Divided by 16 because it '*gives a good noise reduction while maintaining a reasonable rate of convergence*' [47]

RFC	Media Type	RTP Payload Format
RFC 5691	audio	RTP Payload Format for Elementary Streams with MPEG Surround Multi-channel
RFC 5219	audio	A more Loss-tolerant RTP Payload Format for MP3 Audio
RFC 3640	video/audio	RTP Payload for Transport of MPEG4 Elementary Streams
RFC 3016	video/audio	RTP Payload Format for MPEG4 Audio/Visual Streams
RFC 2250	video/audio	RTP Payload Format for MPEG1/MPEG2 Video

Table 2.26: A sample list of RFC for RTP Payload Media Types

2.4.2.2 Analysing Sender and Receiver Reports

Both, senders and receivers, benefit from the information reported by SR and RR RTCP packets. Sender and receiver can react to the information to improve QoS, e.g., sender may modify its transmission and/or determine Round Trip Times. Receivers can use RTCP RTP/NTP to implement inter-stream synchronisation if both streams originate from the same source and, thus, share a wall-clock NTP time [47].

Jitter indicates network congestion whereas packet lost indicates either severe congestion or noise congestion. The two parameters are related due to jitter being a congestion indicator often causing packet lost [47].

2.4.3 RTP Payload for MPEG Standards

There are numerous RFCs defining specific RTP payloads for multimedia data although this section focuses on those related to MPEG standards. There are two especially for audio, RFC 5691 and RFC 5219, and three for video, RFC 3640, RFC 2250 and RFC 3016, summarised in Table 2.26. In this section, the different RTP payload type are described, whereas Section 2.4.3.1 is especially relevant to the prototype whereas RTP payload types are not applied.

In Fig. 2.33, an MP2T packet is shown within an RTP packet (with special attention to MP2T time related values) and also the mapping between the RTP timestamp value and the RTCP NTP wall-clock value within the RTCP SR packet.

2.4.3.1 RFC 2250: RTP Payload for MPEG-1/MPEG-2

Conveying MPEG-1/MPEG-2 using a specific RTP payload accomplishes two main objectives. Firstly, it provides compatibility between MPEG systems and second, it supports compatibility with other RTP conveyed media streams. RFC 2250 defines two different encapsulation methods to carry MPEG-1 and MPEG-2 that facilitate each approach, conveying MP2T/MP2P or ES [48].

There are two payload formats, the first encoding, MPEG-1 system stream packets (MP2T or MP2P); and the second encoding ES directly within the RTP payload. The former provides maximum compatibility between MPEG systems and the latter maximum interaction between

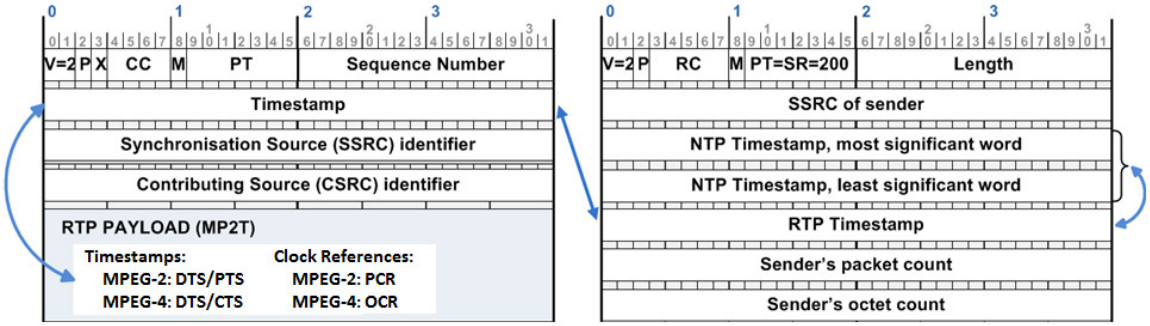


Figure 2.33: MP2T conveyed within RTP packets and the mapping between RTP timestamp with the RTCP SR NTP wall-clock time

RTP Field	Meaning when RFC 2250 Payload with MP2T
Payload Type	Indicates type of data conveyed in the payload. MPEG-1 Systems Streams, MPEG-2 PS or MPEG-2 TS. For RTP Payload Type the value is 33 [48]
Timestamp	'*32 bit 90KHz timestamp representing the target transmission time for the first byte of the packet payload*' [48]

Table 2.27: RTP Header Fields meaning when RFC 2250 payload is used conveying MP2T packets

different RTP conveyed media streams [48].

Encapsulation of MPEG System and MP2T/MP2P An RTP packet may carry multiple MP2T, MP2P or MPEG-1 system packets. As described, the size of MP2T is fixed at 188 bytes, thus, the number of MP2T packets within a RTP packets equals the RTP payload length divided by 188 bytes. By contrast, the unpredictable size of MP2P and MPEG-1 systems packets makes the number of packets unknown.

The RTP header for MP2T/MP2P encapsulation has its own fields which have a dedicated value as defined by RFC 2250 payload type and shown in Table 2.27 fields *payload type* and *timestamp*.

Encapsulation of MPEG Elementary Streams As outlined above, elementary streams (ES) may also be conveyed directly within RTP packets e.g., MPEG-1 and MPEG-2 audio and video. Due to the lack of systems headers coded within the stream this method is more impacted by packet-loss. Thus, some information should be added in the RTP payload to facilitate some recovery techniques at the application layer [48].

In Table 2.28 the RTP fields and their meaning in the case of ES are explained.

An audio ES needs a special header, MPEG audio-specific header, to be located after the RTP fixed header. Similarly, video ES also require a special header, MPEG video-specific header, and in case of MPEG-2 ES also an video-specific auxiliary header is also needed [48].

RTP Field	Meaning when RFC 2250 Payload and ES
Payload Type	'*MPEG video or audio stream ID*' [48]
Timestamp	'*32 bit 90KHz timestamp representing presentation time of MPEG picture or audio frame. Same for all packets that make up a picture or a audio frame. May not be monotonically increasing in video stream if B pictures present in stream. For packets that contain only a video sequence and/or GOP header, the timestamp is that of the subsequent picture*' [48]

Table 2.28: RTP Header Fields when RFC 2250 payload is used for transporting ES streams

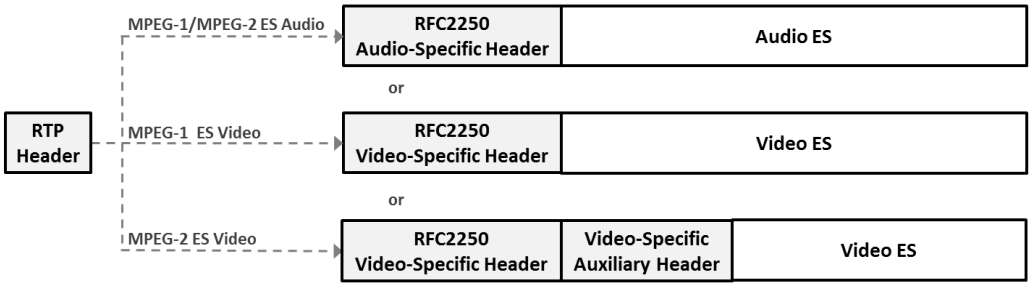


Figure 2.34: High Level RFC 2250 payload options for ES payload

In Fig. 2.34 the three options are shown, with the specially inserted header, just after the RTP Header, in each of the scenarios.

MPEG Video Elementary Streams The minimum size of a RTP payload is 261 bytes therefore the RTP payload should at least contain the largest ES header, with *quart_matrix_extension() extension_data()*. Fragmentation of a large picture into packets is applied following some rules affecting the location of *video sequence header*, *GOP header* and *picture header* when they are present in the RTP payload. First, the *video sequence header* shall always be at the start of the RTP payloads; second, the *GOP header* shall be at the beginning of a RTP payload or behind the *video sequence header* and, finally, the *picture header* shall be at the start of a RTP payload or following a *GOP header* [48].

Particular case is the *video sequence header* which is encoded multiple times in the video stream to facilitate channel switching between MPEG programs.

Slices play a special role as a *'unit of recovery from data loss and corruption'* [48]. The only requirement for its fragmentation is that the slice data shall be located behind the ES header at the beginning of a RTP payload or following other slices within the RTP payload. This ensures that in case of packet lost, the next slice can be rapidly found at the beginning of the following RTP packet.

Table 2.29 lists all fields within the MPEG Video-specific Header common to MPEG-1 and MPEG-2 whereas the fields within the MPEG-2 Video-specific Extension Header are described in Table 2.30.

Field	Bits	Description
MBZ	5	Unused. Set to zero for future use
T	1	MPEG2 specific header extension present
TR	10	Temporal-reference
AN	1	Active N bit for error resilience
N	1	New picture header
S	1	Sequence-header present
B	1	Beginning-of-slice
E	1	End-of-slice
P	3	Picture-Type
FBV	1	Full_pel_backward_vector
BFC	3	Backward_f_code
FFV	1	Full_pel_forward_vector
FFC	3	Forward_f_code

Table 2.29: MPEG Video-specific Header from RFC 2250 [48]

MPEG Audio Elementary Streams An RTP packet may convey multiple entire audio ES or a large audio ES can be conveyed via multiple RTP packets. *'For example for Layer-II MPEG audio sampled at a rate of 44.1 KHz each frame would represent a time slot of 26.1 ms. At this sampling rate if the compressed bit-rate is 384 kbs then the average audio frame would be 1.25 Kbytes'* [48].

'For either MPEG1 or MPEG2 audio, distinct PTS may be present for frames which correspond to either 384 samples for Layer-I, or 1152 samples for Layer-II or Layer-III. The actual number of bytes required to represent this number of samples will vary depending on the encoder parameters' [48].

2.4.4 RTP issues with Internet Media Delivery

The rationale to move away from RTP to HTTP Adaptive Streaming in Internet is outlined in the three following reasons [49]:

- RTP with UDP does not often perform well in best-effort Internet due to its varying and non ideal network conditions
- The use of dynamic port numbers by RTP makes Firewall/NAT traversal difficult. Various research efforts have tried to solve this issue such as the use of tunnelled RTP over TCP/RTSP.
- The one to one RTP media sessions to clients makes scalability an issue in large systems. Multicast solves the issue in IPTV systems but multicast is not possible in Internet

RTP is used with UDP for real-time communications although if real-time delivery is not required, HTTP and TCP are best suited, which explains the move to these for Internet Radio

Field	Bits	Description
X	1	unused
E	1	Extensions present
f_[0,0]	4	forward horizontal f_code
f_[0,1]	4	forward vertical f_code
f_[1,0]	4	backward horizontal f_code
f_[1,1]	4	backward vertical f_code
DC	2	intra_DC_precision
PS	2	picture_structure
T	1	top_field_first
P	1	frame_predicted_frame_dct
C	1	concealment_motion_vectors
Q	1	q_scale type
V	1	intra_vlc format
A	1	alternate scan
R	1	repeat_first_field
H	1	chroma_420_type
G	1	progressive frame
D	1	composite_display_flag

Table 2.30: MPEG Video-specific Header Extension from RFC 2250 [48]

delivery over Internet.

2.4.4.1 Issues relating RTP over UDP with NAT/Firewalls

As RTP is carried over UDP, it creates Network Address Translation (NAT) and firewalls problems for multimedia delivery over IP Networks, such as VoIP which use this protocol. The issue comes from the SIP and SDP media session connections and the RTP/UDP media traffic delivery.

NAT devices provide transparent routing to hosts by mapping the private network unregistered IPs to a public network registered IPs [50].

The NAT problems arise because of the modification of IP addresses, changed from private to a public IP addresses. When this happens, the response from the media server is to drop the packet at NAT because there is a mismatch between the initial out-coming address, from NAT to media server, and the incoming address, from the media server to NAT. Fig. 2.35 shows the example of the issue detailed [50]. The figure shows the communication timeline and the point where the packet ultimately is dropped by the NAT because the IP address and ports don't match. This issue has been investigated and many solutions over the time have been deployed, but still it is a drawback to the use of RTP over UDP media delivery. There is research performed using NAT traversal techniques but they are out of the scope of this thesis [50] [51].

A Firewall is a network element which protects a sub-network from undesired network

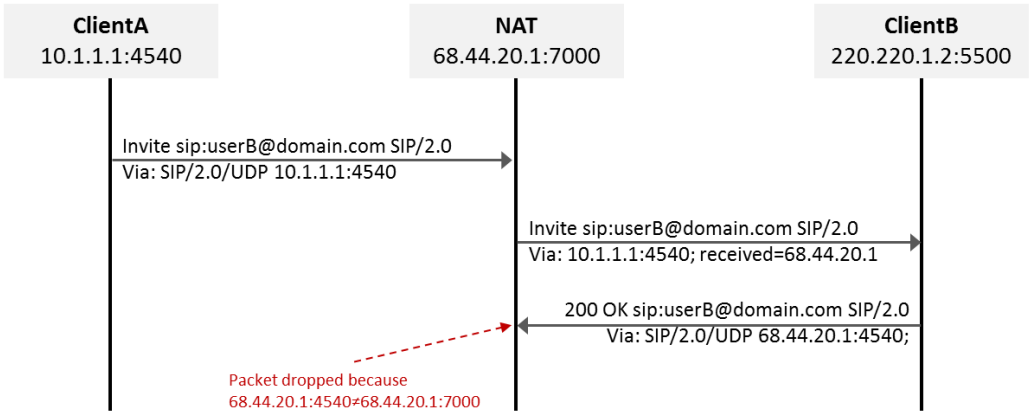


Figure 2.35: Example of connection media session highlighting NAT problems [50]

traffic. It is located between the sub-network and the Internet. It protects the sub-network from incoming traffic and prevents network elements inside the sub-network to access unwanted services from Internet.

Whilst these are sound reasons for firewall deployment, implementation of such rules has significant impact on RTP traffic. For example firewalls, for security reasons, will also block unsolicited SIP REGISTER requests to register servers and unsolicited SIP INVITE requests to proxy servers [51]. Furthermore, media sessions using Dynamic Random ports are also blocked by firewalls and thus block UDP traffic [50].

Although RTP is a recommended protocol for IPTV (private, well-manage IP networks) and for real-time media delivery, whereas for Internet TV, HTTP Adaptive Streaming is the protocol used for live TV channels over Internet (public non-managed IP networks) for the above reasons.

2.4.5 MMT versus RTP and MP2T

MP2T, although being the media container most widely used in broadcasting technology, does not provide hybrid delivery. Moreover it does not share STC among multiple encoders. RTP, the media delivery protocol described later in detail, delivers independent components, thus, does not provide tools for content file delivery. Finally, no storage format is specified by RTP. In Fig. 2.36 and Table 2.31 a comparison between MMT, MP2T and RTP is presented [46].

MMT is the proposed solution to provide full support to the missing features to provide hybrid media delivery, broadband and broadcast media delivery over NGN [46].

The MMT solution provides media assets QoS management as well as multiplexing of several media components into a single flow. Additionally, it provides media sync based on UTC, multiplexing media assets and buffer management. Further details on timing in MMT is provided in Chapter 3.

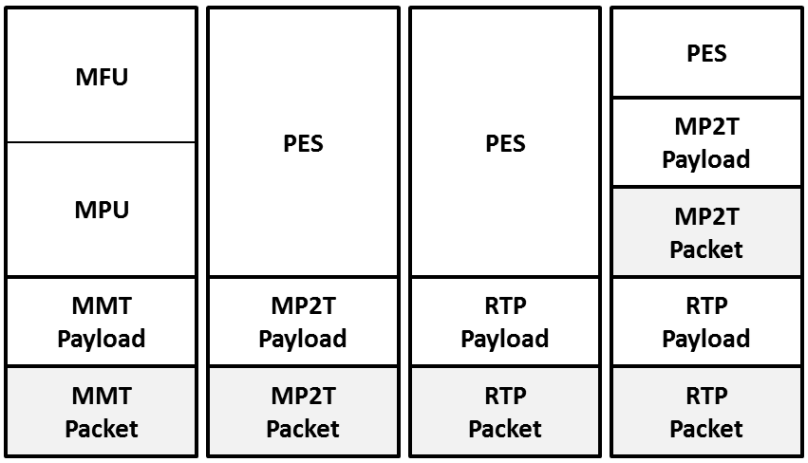


Figure 2.36: MMT protocol stack [46]

Function	MMT	MP2T	RTP
File Delivery	Yes	Partially yes	External
Multiplexing media components and signalling messages	Yes	Yes	No
No multiplexing media components and signalling messages	Yes	No	Yes
Combination of media components on other networks	Yes	No	Yes
Error resiliency	Yes	No	External
Storage format	Partially yes	Partially yes	No

Table 2.31: Functional comparison of MMT, MP2T and RTP [46]

2.4.6 HTTP Adaptive Streaming

2.4.6.1 HTTP Adaptive Streaming

As outlined above, one of the latest media delivery protocol is HTTP Adaptive Streaming. In this section, the focus is on MPEG-DASH, the independent MPEG Standard. In Table 2.32 the main characteristics of HTTP Adaptive protocols are listed. Table 2.33 presents a comparison between two HTTP Adaptive Protocols, HLS and MS-SSTR.

Dynamic Adaptive Streaming over HTTP is the protocol preferred for streaming services, instead of the traditional RTP and RTSP protocols. This is for a variety of reasons including[52]:

- HTTP legacy: HTTP is the principal multimedia delivery protocol used in Internet. It avoids the NAT and Firewall traversal issues associated with UDP as it is based on the widely used TCP/IP protocol providing reliability and deployment simplicity. The use of existing HTTP servers and HTTP caches to deliver media via a Content Delivery Network (CDN) also provides a ready infrastructure.

RTSP	HLS	MS-SSTR	RTMP
IPTV	Internet TV	Internet TV	Internet TV
RTP Packets	HTTP Segments	HTTP Fragments	RTMP chunks
IETF	Apple	Microsoft	Adobe Flash
TCP	TCP	TCP	TCP
MP2T	MP2T	MPEG4 part 14	Multiple
Stateful	Stateless	Stateless	Stateless
No handshake	No handshake	No handshake	No handshake

Table 2.32: HTTP Adaptive Protocols Characteristics [53]

	HLS	MS-SSTR
Company	Apple	Microsoft
Media Server	HTTP Server	IIS Extension
Information File	Index File	Client and Server Manifest File
Format Information File	Index File format M3U8	Manifest File XML format
Video Codec	H.264	H.264
Audio Codec	MP3 and AAC	AAC
Media Container	Each segment stored as MP2T	MP4 virtual fragmented file
Media Divided into	Media segments	Fragments

Table 2.33: Comparative HLS and MS-SSTR solutions

- Client-driven: It provides total client control of the streaming sessions by allowing the client to choose the content rate to suit the available bandwidth and device. It seamlessly changes the content rate to suit the available bandwidth.
- Allows CDN to be use as a common delivery platform for the fixed and mobile convergence.

The adoption of Dynamic Adaptive Streaming over HTTP provides '*an efficient and flexible distribution platform that scales to the rising demands*' [52]. The main benefit is that traditional RTSP streaming is based on a stateful[1] protocol whereas HTTP is a stateless protocol, whereby an HTTP request is a '*standalone one-time transaction*' [52], which facilitates scalability. MPEG-DASH is the HTTP Adaptive Streaming over HTTP chosen by 3rd Generation Partnership Project (3GPP)[2] to support multiple services such as On-demand streaming, linear TV including live media broadcast and time-shift viewing with network PVR [52]. The following section reviews MPEG-DASH in detail.

2.4.6.2 MPEG-DASH

MPEG-DASH is the ISO/IEC 23009 part 1 Standard for Adaptive HTTP Streaming. It is based on the HTTP application protocol, the media delivery is guided by the client to provide

[1] Server that retains state information about client's request
[2] http://www.3gpp.org/about-3gpp

adaptive media delivery to end-users adjusted to the client's changing requirements.

MPEG-DASH's main tool to provide such adaptive functionality is the Media Presentation Description (MPD) file. This XML-based file provides the HTTP client with the information required to select the media files/streams most appropriate to the user's capabilities. Therefore, the client guides/pulls the media delivery from the server.

The benefits of MPEG-DASH include the availability to perform well under varying bandwidth conditions [54] often experienced in the Internet. As discussed previously, it solves the NAT and Firewall traversal problems, the main issues with RTP media delivery. MPEG-DASH provides a '*flexible and scalable deployment as well as reduced infrastructure costs due to the reuse of existing Internet infrastructure components*' [54].

MPEG-DASH works with HTTP/1.1 but the performance of MPEG-DASH working with HTTP/2.0 has also been studied, in particular focusing on the protocol overhead and the performance (under different round trip times) [54].

MPEG-DASH is the subject of significant research due to the rapid growth in Internet video streaming. For example Scalable Video Coding (SVC) extensions have been integrated to the MPEG-DASH Standard evaluating its implementation [55] and the quality of MPEG-DASH is evaluated when media streaming is switched from one end-device to another [56]. Finally, one of the most popular media players, VLC, used in this thesis has been extended for MPEG-DASH play-out, as shown in [57].

MPEG-DASH provides additional, flexible and extensible features enabled for different future uses, such as [58]:

- Switching and selectable streams: The MPD file provides the means to select from different streams. E.g., different audio or subtitles for the same video or different video streams (i.e., from different camera angles) from the same event.
- Ad insertion: Adverts can be added between periods or segments.
- Compact manifest: A compact MPD file can be created by using segment address URL.
- Fragmented manifest: MPD file can be sent to the client in separate parts which are downloaded in different steps.
- Segments with variable durations: Duration of the segments duration is variable and one segment can inform about the next segment's duration.
- Multiple base URLs: The same media content could be accessible from different URLs (different media servers or CDNs).
- Clock-drift control for live sessions: UTC information could be added in each segment.
- SVC and Multiview Video Coding (MVC) support: the MPD facilitates decoding information dependencies which are used by multilayer coded streams.

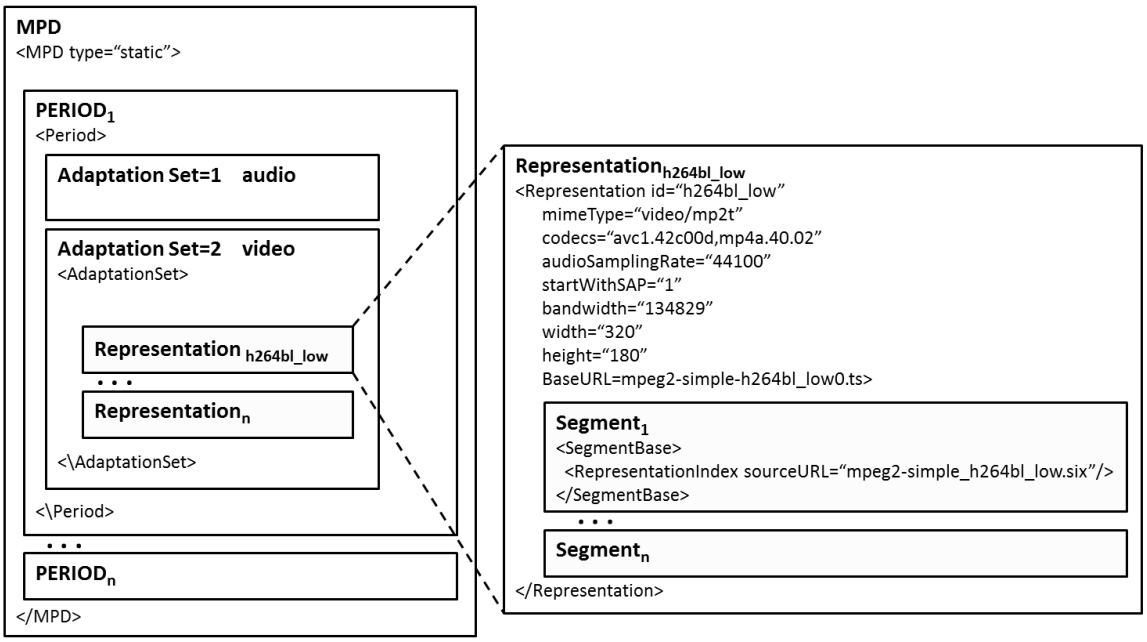


Figure 2.37: MPD file example

- A flexible set of descriptors: Descriptors are used to provide the receiver with the information required to perform the media decoding process.
- Sub-setting adaptation sets into groups: AdaptationSet provides the means to group the media content under the author's consent.
- Quality metrics for reporting the session experience: The client monitors and reports back, using well-defined quality metrics, information about the session experience to a reporting server.

The main factors considered by the client are hardware, network connectivity (bandwidth) or decoding capabilities. Thus, the client via the MPD file selects the media file best suited for the media session delivery. The MPD file contains the URLs of the available media segments in the MPEG-DASH server.

An MPD file type can be Static, for VoD, or Dynamic, for live media delivery. The MPD type sets the fields requirements within the MPD file.

The main MPD elements are Media Presentation (*MPD*), *Period*, *AdaptationSet*, *Representation* and *Segments*. The *MPD* contains the media delivery general information and includes the information to splice the media content. An MPD file is divided into periods which indicate a time frame. Within periods the *AdaptationSet* wraps the multiple representations of the media type/content. The *Representation* describes the media of a specific representation and contains the media segments in the specific representation. An example of an MPD file can be

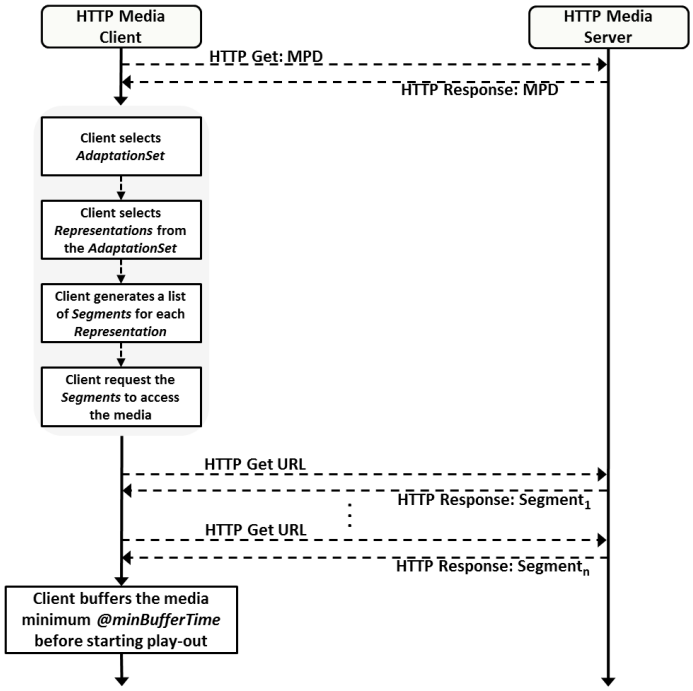


Figure 2.38: MPEG-DASH Client example from [59]

found in Fig. 2.37.

An example of MPEG-DASH behaviour is drawn in Fig. 2.38. The client requests the MPD file via a *HTTP Get* and the server replies sending the MPD file via a *HTTP Response*. From there the client selects the *AdaptationSet*, then *Representation* within the *AdaptationSet*. Then, the client generates a list of *Segments* for each *Representation*. Finally, the client requests the *Segments* to access the media which will be delivered via HTTP [59].

Once the client receives the media segments, it buffers and then the media play-out begins. The client informs the HTTP media server when it wants to stop the media delivery.

2.5 Summary

This chapter commenced by briefly discussing the terms QoS and QoE as ultimately, the thesis is all about providing an enhanced user experience. It then covered in detail, all of the principal components that collectively provide a media content and delivery architecture. In particular, it covered the following key areas.

2.5.1 Media Delivery Platforms

There are two main platforms for media delivery: broadcast and broadband. The former includes DVB cable, satellite and aerial whereby media is broadcasted to clients. The latter

includes two systems, IPTV or Internet TV. IPTV is based on multicasting to clients using a private well-managed network whereas Internet TV is delivered via unicast to clients via the public Internet network, thus raising a range of QoS issues. Regarding IPTV, the chapter described the media content delivered via the platform, the principal functions and services including the application, service, transport and media functions. It outlined the IPTV main structure and gave a brief introduction to the communication protocols used by IPTV. Regarding Internet TV, the chapter outlined the media codecs, the media delivery protocols used and the principal media delivery protocol, Adaptive HTTP Streaming, and in particular MPEG-DASH. Finally, this section provided an overview of HbbTV covering its main HbbTV structure, media formats and protocols used, in particular RTSP and SDP. HbbTV provides a unique client-side platform which integrates media received via both media delivery platforms, broadcast and broadband.

2.5.2 Media Containers

The media containers studied to packetise media streams in the thesis are: MPEG-2 TS (used in prototype), MPEG-4, ISO BMFF and MP3 (used in prototype). Moreover DVB-SI and MPEG-2 PSI are studied because of the information they provide within MPEG-2 TS and also because they are the DVB tool to transmit services and program information within MP2T streams.

The latest MMT media container is also included as the latest MPEG standard that aims to integrate broadband and broadcast media delivery systems to facilitate media integration at client-side.

2.5.3 Transport Protocols

Finally, this chapter detailed the RTP protocol, as a key real-time transport protocol. It focused on the RTP timestamps, the principal RTP payloads used for MPEG-1, MPEG-2 (RFC 2250) and MP3 (RFC 5219). It also covered the issues relating to use of RTP protocol over UDP with NAT and Firewalls. RTP RET was also briefly described as a solution to packet loss issues in such environments.

In summary, this chapter dealt with the key infrastructure components that collectively facilitate media encapsulation and delivery. The next chapter focuses entirely on the timing and synchronisation of multimedia, and sets the context for the specific contribution of this research, namely, media synchronisation on a single device from disparate sources and delivered via different platforms.

Chapter 3

Multimedia Synchronisation

The previous chapter detailed the key infrastructure components that collectively facilitate media encapsulation and delivery thus setting the context for the thesis. This chapter examines the core thesis issue of multimedia synchronisation.

As synchronisation is closely related to timing, the chapter firstly reviews how computer clocks typically operate, what issues can arise and how this can impact on multimedia. It then reviews media sync types, sync thresholds, and time protocols such as Network Time Protocol (NTP) and Precision Time Protocol (PTP), as well as time sources such as Global Navigation Satellite Systems (GPS). Following this, it examines a range of multimedia sync solutions and applications including Inter-destination Media Synchronisation (IDMS) and ETSI TS 102 823 (solution used by HbbTV). Thirdly, synchronisation within MPEG is examined in detail, including, MP2T timelines, clock references and timestamps, MPEG-2 part 9, the extension for Real Time Interface for system decoders and ETSI 102 034 MPEG-2, and timing reconstruction within MP2T transport based on DVB services over IP Networks. Finally, this chapter also describes the timelines of other MPEG standards that are not core to the thesis implementation but are relevant in overall context of thesis contributions. These include MPEG-4, ISO, MPEG-DASH and MMT. In Appendix C summarizes all clock references and timestamps in MPEG-1, MPEG-2 and MPEG-4.

The relevant sections of this chapter to the prototype are thus MPEG-2 part 1, MP3, DVB-SI and MPEG-2 PSI whereas the areas MPEG-4 part 1, ISO, MPEG-DASH and MMT are described to provide a general view of the different timelines implementations in MPEG standards.

3.1 Clocks

Clocks play a key role in media sync. Ridoux describes three clock purposes. Firstly, to establish the time of the day (ToD), secondly, to order events, and thirdly, to measure time between

events [60].

Clocks provide the two related services of time and timing. Time relates to the commonly accepted time-of-day that is based on the widely accepted time standard, Coordinated Universal Time (UTC). Timing relates to the frequency at which a clock runs. Both concepts are important in that certain applications may require one or the other or both. E.g., for timestamping of events, time is important, whereas the challenge of matching a decoder to an encoder relates to timing.

Two concepts define a clock, frequency and resolution. Frequency is the rate at which a physical clock's oscillator operates, in other words, the clock's rate of change. A clock's resolution is *'the smallest unit by which the clock's time is updated. It gives a lower bound on the clock's uncertainty'* [61]. Resolution is also known as precision.

Computer clocks have varied precision values. One example of the popular Windows operating system is the precision of Microsoft's Windows 7 OS which can be as coarse as 15.625ms [62]. Moreover, Linux operating systems have different precision values, ranging from 1 us to ms. As an example, Minix OS presents a precision of 16 ms [63] other Linux systems such as FreeBSD and DragonFlyBSD, can be up to 1ms or better [64]. In the context of this project, clock resolution is an important issue as timestamps need to be fine enough to facilitate precise synchronisation.

3.1.1 Delivering Clock Sync (NTP/GPS/PTP)

There are, many sources of absolute time, each with their own characteristics. Simple quartz crystals found on consumer electronics work reasonably well but their skew rate can be ±100ppm leading to increasing offsets of seconds/day. Oven and temperature compensated clocks are better but more expensive. Atomic clocks are extremely accurate but very expensive. The use of Global Navigation Satellite System (GNSS), such as GPS, Glonass and Galileo provide access to atomic clock level accuracy though signal strength can be an issue. The other issue with time is how to distribute it across a network. Again, various solutions exist but the most common entail the use of protocols such as NTP and PTP.

GNSS systems provide time and location via multiple earth orbiting satellites. Many modern receivers can utilise signals from various constellations. Such systems typically have their own time references, e.g., GPS-time which is the time from its epoch, the 6^{th} of January of 1980 at midnight. GPS time does not include leap seconds, thus, it is ahead of UTC [65].

Computer systems connected over IP Networks are typically synchronised via NTP. Therefore, media servers and media receivers synchronised via NTP are synchronised to the same epoch. Theoretically, NTP can facilitate a precision as high as 232 picoseconds, as timestamps in NTP are *'64 bit unsigned fixed-point numbers with the integer part in the first 32 bits and the fraction part in the last 32 bits'* [66]. Variable latency and dynamic networks can however limit synchronisation accuracy values to the order of milliseconds across WANs and approximately 1ms over LANs.

NTP is a robust protocol. The time reference of a host is obtained from multiple NTP time servers. These time reference responses, after statistical analysis, provide an improved estimation of true time. This is the key to its robustness as, due to multiple time sources, the protocol can adapt in the event of an unreachable server [66].

NTP host and server typically operate in client/server mode. The host periodically requests time from the server, and servers respond to every request. The communication between host and servers is achieved via NTP packets transmitted via UDP/IP.

The host and server request and response, respectively provide four timestamps, namely, origin ($t1$), receive ($t2$), transmit ($t3$) and destination ($t4$) timestamp. These timestamps provide enough information to allow the host to determine its time difference from the server, presuming symmetric networks. This latter presumption introduces significant noise.

NTP is quite a complex protocol. Therefore, for computer systems that only need to synchronise loosely to an external time source, the Simple Network Time Protocol (SNTP) was developed. It is a simplified and fully compatible version of NTP. NTP and SNTP share the same NTP timestamps formats, message packet header, and both use UDP over IP to deliver their protocol packets [67].

The more recent alternative to NTP, PTP is mainly designed for use in well managed Ethernet and multicast-capable networks and it is designed with specific PTP-aware hardware to provide sub 1μs accuracy between the nodes of a distributed system. It is based on a master-slave configuration. PTP uses two-way message exchange mechanism similar to NTP to calculate offset between slave and master [68].

There is outgoing work to augment the information provided via SDP to facilitate multimedia synchronisation. IETF Internet Standard [69] proposes to share synchronisation media sources information, such as synchronisation protocol and sources (e.g., NTP, PTP, GPS, Galileo reference or local) and parameters used at the media source by using SDP.

3.1.2 Clock signalling

As detailed above the use of NTP and PTP allow accurate time to be distributed across a network. What is also increasingly important is a mechanism to communicate clock related characteristics between media. An IETF Standard Track to provide *RTP Clock Source Signalling* has been recently published (June 2014). It aims to provide multimedia sessions with information about the timestamping media clock sources via SDP signalling [69]. Not used in the thesis although it is relevant as a tool to provide clock information among media sessions.

RFC 7273 standard provides for added fields in SDP to inform receivers about the clock used at encoder's side in the timestamping process. This is performed at session (information related to the all session), media (information related to a media stream) and source (information related to a media source) level. The main structure of the information is the following:

- Session level → a=ts-refclk:<clksrc>

v=0
o=jdoe 2890844526 2890842807 IN IP4 192.0.2.1
s=SDP Seminar
i=A Seminar on the session description protocol
u=http://www.example.com/seminars.sdp.pdf
e=j.doe@example.com (Jane Doe)
c=IN IP4 233.252.0.1/64
a=recvonly
a=ts-refclk:ntp=/traceable/
m=audio 49170 RTP/EVP 0
m=video 51372 RTP/AVP 99 a=rtpmap:99 h263-1998/9000

Table 3.1: Example Clock Signalling at Session Level in Figure 2 from [69]

v=0
o=jdoe 2890844526 2890842807 IN IP4 192.0.2.1
s=SDP Seminar
i=A Seminar on the session description protocol
u=http://www.example.com/seminars.sdp.pdf
e=j.doe@example.com (Jane Doe)
c=IN IP4 233.252.0.1/64
t=2873397496 287340496
a=recvonly
a=ts-refclk:local
m=audio 49170 RTP/EVP 0
a=ts-refclk:ntp=203.0.113.10
a=ts-refclk:ntp=198.51.100.22
m=video 51372 RTP/AVP 99
a=rtpmap:99 h263-1998/9000
a=ts-refclk:ptp=IEEE802.1AS-2011:39-A7-94-FF-FE-07-CB-D0

Table 3.2: Example Clock Signalling at Media Level. Figure 3 in [69]

- Media level → a=ts-refclk:<clksrc>
- Source level → a=ssrc:<ssrc-id> ts-refclk:<clksrc>

The clock signalling defined at media and source level override the session level defined values. There are multiple fields but the key ones are:

- *timestamp-refclk="ts-refclk:" clksrc CRLF*
- *clksrc= ntp/ptp/gps/gal/glonass/local/private/clksrc-ext*
- *clksrc-ext = clksrc-param-name clksrc-param-value*
- *clksrc-param-value = ["=" byte-string]*

There are different ways to use SDP clock signalling [69], in Table 3.1 an example at session level is found. Table 3.2 presents media level and finally, Table 3.3 presents source level.

v=0
o=jdoe 2890844526 2890842807 IN IP4 192.0.2.1
s=SDP Seminar
i=A Seminar on the session description protocol
u=http://www.example.com/seminars.sdp.pdf
e=j.doe@example.com (Jane Doe)
c=IN IP4 233.252.0.1/64
t=2873397496 287340496
a=recvonly
a=ts-refclk:local
m=audio 49170 RTP/AVP 0
m=video 51372 RTP/AVP 99
a=rtpmap:99 h263-1998/9000
a=ssrc:12345 ts-refclk:ptp=IEEE802.1AS-2011:39-A7-94-FF-FE-07-CB-D0

Table 3.3: Example Clock Signalling at Sources Level. Figure 4 in [69]

3.2 Media synchronisation

There are multiple factors affecting the perception of audio-video synchronisation in TV. These include the acquisition equipment (audio and video characteristics), program composition (close-up image, head and shoulders or wide-shot), production equipment, production processing, reproduction equipment and perception processing (user's distance from screen) [70].

Multimedia sync relates to the synchronisation of time-related varied media, a requirement that is made more challenging when media is delivered over non deterministic packet based networks such as the Internet. The different media types can include, video, audio and still picture, each of which may use different formats namely, MPEG-2 and MPEG-4 video format, or MP3, AAC and WMA audio format. The media type and its format can be fundamentally represented as Media Access Unit (MDU) or AU, the smallest timed media unit.

Multiple factors combine to affect media sync, from the source, through the IP Network, to the receiver. In Table 3.4 all factors are summarized whereas in [73] related work is presented, expanding the parameters that could affect temporal relationships.

The parameters related to the network are Network Delay and Network Jitter. Network Delay is the delay of the MDUs within the network, and the Network Jitter, which is the variation of the Network Delay.

The parameters related to both encoder and decoder's clock differences are described by Clock Offset, Clock Skew and Clock Drift. The Clock Offset is defined as the time difference between two clocks. The Clock Skew is the frequency difference, defined as rate of change of offset caused by clocks' imperfections and the Clock Drift is the rate of change in frequency over time induced by factors such as the temperature, pressure, and voltage crystal ageing.

Finally, End-system Jitter is caused by the various tasks performed at encoder and decoder for media encoding and decoding can also be present.

To overcome all these factors that potentially affect media sync, different tools and tech-

	Cause	Definition	Caused by
Network	**Network Delay**	Delay one packet experiences from the source, through the network, to the receiver	Network load/traffic (congestion), network devices latency and serialization delay
	Network Jitter	Variation in delay	Network varying conditions (e.g., load, traffic, congestion...)
End-System	**End-system jitter**	Delay at the end-systems caused by the task of packetisation/depacketisation AUs through protocols in different layers, encoding/decoding media, OS applications, jitter buffers, display lag, etc	System load/hardware
Clock	**Clock Offset**	'*Difference in clock times*' [72]	Initialisation offset
	Clock Skew	'*First derivative of the difference in clock times*' [72]. Frequency difference	Imperfections in clock manufacturing process
	Clock Drift	'*Second derivative of the difference in clock times*' [72]. Change in frequency over time	Temperature, Pressure, Voltage, Crystal ageing, effect over time causing clock drift

Table 3.4: Parameters affecting Temporal Relationships within a Stream or among multiple Streams [71]

Sync Type	Sub-type	Description
Intra-media Sync		Sync within a single media stream
Inter-media Sync	Lip-sync	Video and audio sync
	IDMS	Inter-Destination Media Sync
	IDES	Inter-Device Sync
	Point-Sync	Two sync points: start and end
Multimedia sync		Intra and inter media sync altogether
Hybrid Sync		Require intra and inter-media sync (HbbTV)
Interactive Sync		Sync with full user's interaction
Adaptive Sync		Adaptive time presentation adapted to network conditions

Table 3.5: Media Sync classification. Sync types and sub-types

niques are used. In the following sections, firstly the different media sync types are described, secondly, synchronisation methods are discussed, and thirdly sync aspects relating to MPEG standards are reviewed.

3.2.1 Multimedia Sync Types

In Table 3.5, various media sync types are listed. The two main types, inter and intra-media sync, are further described in the following subsections. Merging these two types introduces the concept of simultaneous intra and inter-media sync play-out.

There are another three sync groups that relate to external factors. For example, Interactive

Sync involves sync with user's interaction sync whereas Adaptive Sync adapts sync media play-out to network conditions.

One of the latest categories defined is that of Hybrid Sync [74]. It refers to media sync required for integrating media delivered separately over broadband and broadcast platforms. This sync class requires both intra and inter-media sync for both initial sync (inter-media sync) and continuous sync (intra-media sync).

3.2.2 Intra-media Synchronisation

Intra-media Sync is required to maintain the relationship between consecutive MDUs. Within the MPEG standards, it maintains sync among all MDUs within an MP2T stream and it is performed by means of clock references. In sub-section 3.7.2 clock references are described in detail. Essentially they are the tools used to reproduce the encoder's clock at the decoder. Fig. 3.1 shows how intra-media and inter-media sync relates to the MDUs from the two distinct though logically related and how clock skew results in a cumulative sync error. For illustrative purposes, the skew is exaggerated such that whilst both streams are supposed to generated packets at the same rate, MediaStream_1 actually generates packets every 15ms and MediaStream_2 generates packets every 20ms, which will greatly impact on the QoE at user side during the play-out.

3.2.3 Inter-media Synchronisation

Inter-media sync relates to the temporal relationship between MDUs from different media streams. The most popular and clear example is the sync between a video and its audio at the play-out. Video and audio, although multiplexed within the same MP2T stream within MPEG standards are conveyed in two different media streams. As such the time relationship between them relates to inter-media sync. This particular scenario is called Lip-sync. Note that in contrast to MPEG, audio and video delivered using RTP/UDP such as WebRTC video conferencing are carried in completely separate streams and this inter-media sync presents greater challenges.

From Fig. 3.1, it is seen how inter-media sync time aligns MDUs from different media streams. Fig. 3.1 also describes how a slow but constant intra-media sync deviation (skew) affects the play-out media sync by causing a cumulative misalignment between MDUs from different media streams even though each media stream can be perfectly reproduced at receiver when it is a independent play-out.

3.2.3.1 Types Inter-media Synchronisation

Inter-media sync can be further classified depending on other factors, such as media sources, end-devices and end-user applications. On one hand, when trying to sync different media sources, it is referenced as Multi-source Sync. On the other hand, when trying to sync the media

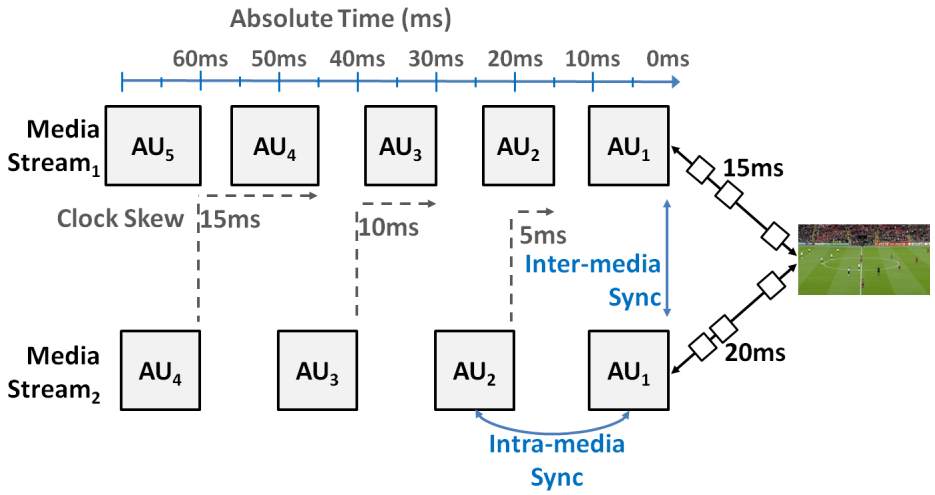


Figure 3.1: Intra and Inter-media sync related to AUs from two different media streams. MediaStream_1 contains AUs different length and MediaStream_2 has AUs constant length

Time	Methods which have end-systems synchronised or not within a network
Location	Methods are performed at the source or the receiver
Method	To modify the generation and presentation speed of MDUs To add or duplicate MDUs also called stuffing To skip or delete MDUs
Participants	Number of participants in media session

Table 3.6: Synchronisation Methods Criteria [75]

play-out in multiple end-users/receivers it is called Inter-Destination Sync (IDMS). Finally, within this group when trying to sync the media play-out at one end-user but multiple media devices it is known as Inter-Device Sync (IDES).

IDMS refers to the media sync play-out to multiple end-users. This is especially used in multi-play games over the Internet where multiple players playing the same game should have synchronised play-out to guarantee fairness in the game within players.

One of the latest categories included is IDES due to the increase in different type of devices used for media play-out. Together, end-users may watch TV over multiple devices, included mobile devices, and during the play-out they interchange the device used. As an example, watching a TV program on the TV device and switching to a tablet and changing to the TV later on again.

In a different category is Point-Sync. Point-Sync refers basically to sync within two time limits. Sync at the beginning and end of the event. This usually relates to sync media such as subtitle streams where initial and final displaying time synchronisation is needed.

Basic	Source Control	Adding timestamps
		Adding sequence number
		Adding sequence marking
		Adding event information
		Adding source identifiers
	Receiver Control	Buffering techniques to avoid buffering starvation and buffering flooding
Preventive	Source Control	Deadline-based transmission scheduling
		Initial transmission and/or play-out instant calculation
		Interleaving MDUs of different media stream in only one transport stream
		Preventive skips of MDUs
		Preventive pauses of MDUs
	Receiver Control	Change the buffering waiting time of the MDUs
		Insert dummy data
		Enlarge or shorten the silence periods of the streams
Reactive	Source Control	Adjust the transmission rate (timing) changing the transmission period
		Decrease the number of media streams transmitted
	Receiver Control	Reactive skips (eliminations)
		Reactive pauses (repetitions or insertions)

Table 3.7: Synchronisation Methods Classification from [73]

3.3 Synchronisation methods

There are multiple techniques to accomplish synchronisation. The most common criteria are specified in Table 3.6. The relevant factors, once the synchronising method is chosen, are when and where to apply it.

There are three broad categories: Basic Control Techniques, Preventative Control Techniques and Reactive Control Techniques. Within those groups, different approaches can be taken. A more extensive list of methods can be found in Table 3.7.

The Basic Control Techniques involve adding extra information to the MDUs at the source side and buffering control at the receiver side. The Preventive Control Techniques compensate for asynchrony before it happens whereas Reactive Control Techniques react to asynchrony.

3.4 Synchronisation Threshold

The user's QoE is the parameter that will dictate the requirements for media sync. This section will first focus on Inter-media Sync, specifically lip-sync, between a video and its audio stream. Finally, the focus will be on media sync threshold classification for IDMS.

Lip-sync parameters have been widely studied, establishing different thresholds depending on application but all of them agree on the general point that users are less sensitive to audio behind the image, called audio lagging, than the audio before the image, audio leading. One of the possible causes for this observation is that people always perceive the sound after the image

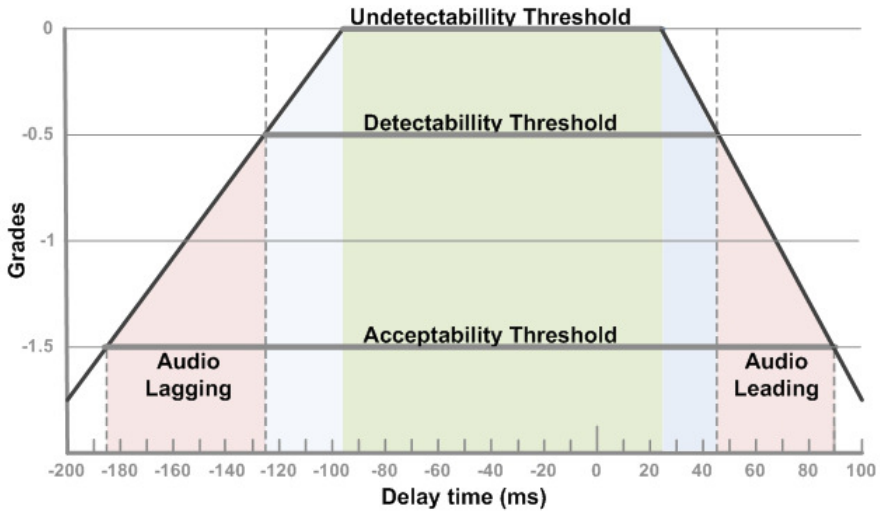


Figure 3.2: Lip-Sync parameters [79]

due to the fact that light travels faster than sound [70]. Light travels $300{\cdot}10^6$ m/s whereas sound is approximately 340m/s.

One classification is defined by the three levels of lip-sync misalignment, unnoticeable sync, noticeable but tolerable and intolerable sync. It is considered tolerable but noticeable if sync levels lie between -80ms to +80ms whereas intolerable sync levels are outside -240ms to +160ms [76].

Another classification is proposed which is even stricter. The acceptable levels of lip-sync are from -60ms to +30ms [77] [78].

Fig. 3.2 shows the levels proposed by the International Telecommunications Union recommendation [79]. In this recommendation, the levels of detectable and acceptable threshold are divided into grades[1]. It shows that sync issues are not detectable between -95ms to +25ms, detectable between -125ms to +45ms and unacceptable outside -185ms to +90ms [79].

QoE sync levels depend on the media, mode and application. In [76] the tighter levels go from 11μs for tightly coupled audio/audio sync to looser sync requirements for audio/pointer sync (-500ms to +750ms).

The sync levels for IDMS are different from the previous lip-sync classifications. The sync levels for IDMS are classified as very high sync (10μs to 10ms), for applications such as networked stereo loud speakers, high level sync (10ms to 100ms), for applications such as multiparty multimedia conferencing, medium level sync (100ms to 500ms) for applications such as second screen sync and, finally, low level sync (500ms to 2000ms) required for social TV [80].

[1] One grade is 45ms for audio leading and 60ms for audio lagging

3.5 Sampling Frequency

ITU-R BT 601-5 recommends the luminance signal frequency of 13.5MHz and each colour-difference signal frequency of 6.75MHz or 13.5MHz [81]. This is the main reason why 27MHz was chosen as a clock reference frequency, and why it appears consistently within MPEG, as described in later sections.

The sampling frequency and the TV line-system has a direct impact on the chosen clock frequency in video encoding because *'In order to sample 625/50 luminance signals luminance signals without quality loss, the lowest multiple possible is 4 which represents a sampling rate of 13.5MHz. This frequency line-locks to give 858 samples per line period in 525/59.94 and 864 samples per line period in 625/50'* [82].

The 625 line-system is used by SECAM and all PAL systems except PAL-M, and is used mainly in Europe, Middle East and the Former Soviet Union. The 525 line-system is used by NTSC and PAL-M and is mainly used in Japan and USA. One relevant data is that and active picture time has 720 pixels in both TV systems (625 and 525 lines) [83]. The frequency should meet the video requirements for both line-systems.

'The importance of the 2.25MHz frequency lies in the fact that 2.25MHz represents the minimum frequency found to be a common multiple of the scanning frequencies of 525 and 625 line systems. Hence, by establishing sampling based on an integer multiple of 2.25MHz (in this case, 6·2.25MHz=13.5MHz), an integer number of samples is guaranteed for the entire duration of the horizontal line in the digital representation of 525/626 line component signals (858 for the 525 line system and 864 for the 625 line system)' [83].

There are other sampling frequencies which differ from 13.5MHz, used in SDTV and HDTV [83]:

- 72MHz is 32 times 2.25MHz
- 74.25MHz is 33 times 2.25MHz (choice for 1125/60 HDTV)
- 81MHz is 36 times 2.25MHz

MPEG-2 has a fixed frequency of 27MHz but MPEG-4 frequency can vary between the values 72MHz, 74.25MHz and 81MHz. In 1125/60 HDTV systems the frequency used is 74.25MHz because *'none of its harmonics interfere with the values of international distress frequencies (121.5 and 2243MHz)'* [83].

The best choice for MPEG-4 is 74.25MHz because it accomplishes a good trade-off between video parameters. The principal ones listed are [83]:

- Practical blanking intervals
- Total data rates for digital HDTV VTRs
- Compatibility with signals of the ITU-R Rec. 601 [81] digital hierarchy

Standard	Frequency (Hz)	Tolerance (Hz)	Tolerance (ppm)
NTSC	3579545.4	±10	±3
PAL (M)	3575611.49	±10	±3
PAL (B, D, G, H, N)	4433618.75	±5	±1

Table 3.8: Specifications for the Colour Sub-carrier of Various Video Formats [84]

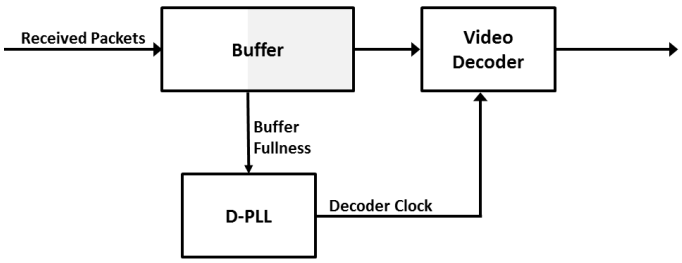


Figure 3.3: Video Synchronisation at decoder by using buffer fullness. Figure 4.1 in [34]

- Manageable signal-processing speeds
- Chroma sampling frequency is 37.125MHz

In Table 3.8 the colour sub-carrier frequencies for different video formats are listed.

3.6 MP2T Timelines

Xuemin Chen describes two possible techniques to achieve video synchronisation. The first technique uses buffer fullness whereas second achieves video sync at decoder through timestamping. The former, as described in Fig. 3.3, uses buffer occupancy to control the D-PLL to provide the Encoder's clock to the Video Decoder. The latter, as described in Fig. 3.4, uses timestamp detection to activate the D-PLL [34]. In this section the technique of clock skew synchronisation via timestamping which is the technique used by MPEG-2 Systems or Transport Streams is further described.

The MPEG-2 Systems Timing model follows the path drawn in Fig. 3.5. A video source provides the input for the MPEG-2 Timing Model where the final output is the constant rate of the reconstructed video [84].

In this Timing model the encoder Compressed Data Buffer (CDB) transforms a variable rate compressed video into a constant rate video compressed output. The decoder's CDB transforms a constant compressed video to a variable rate one. Both CDBs introduce a variable delay in the timing model in both ends of the system, the encoder and decoder's side and, thus, from beginning to end the timing model is considered to have a constant delay [84].

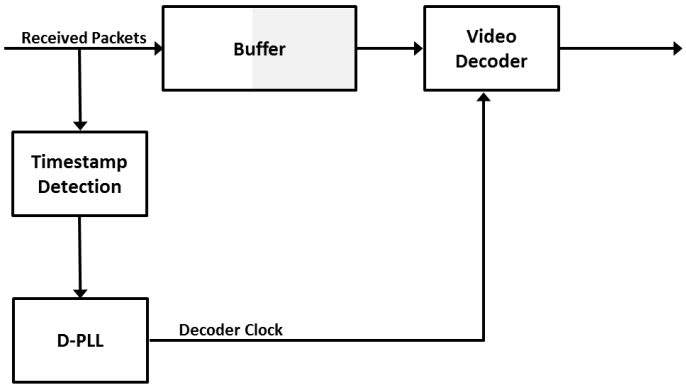


Figure 3.4: Video Synchronisation at decoder through Timestamping. Figure 4.2 in [34]

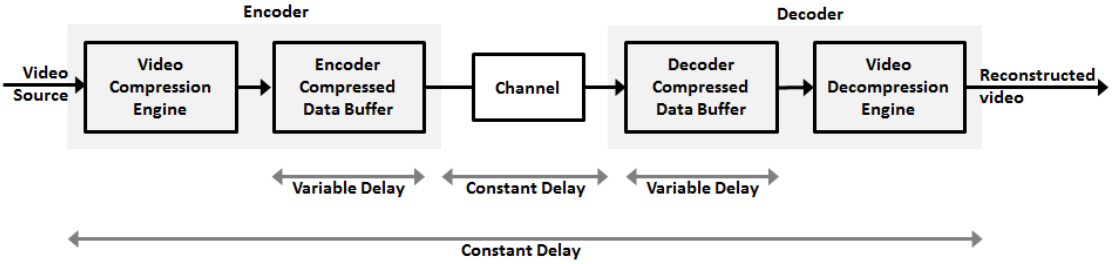


Figure 3.5: Constant Delay Timing Model. Figure 6.5 in [84]

3.6.1 T-STD

Fig. 3.6 shows the video decoding high level diagram with extraction of clock references, PCRs, and timestamps, PTS and DTS. Once the MP2T stream is demultiplexed into its media components the clock references and timestamps are extracted. The PCRs are sent to the D-PLL and DTS/PTS are sent to their respective comparators.

In the centre of the figure the D-PLL is found. There, the decoder's STC is synced with the encoder's PCRs values, making sure the encoder's clock frequency is properly reproduced at the decoder's.

The comparators modules signal when to perform one action or the other. The *Comparator STC/DTS* signals when the video MDU is to be decoded. The *Comparator STC/PTS* signals when the video or audio MDU is to be presented.

There is a difference between the modules for video and audio. This is caused by the nature of both MDU types. In audio the PTS equals DTS (as will be explained later in this chapter) whereas with video this does not apply due to the presence of B-frames. In Fig. 3.6 the module *Frame Reorder Buffer* receives the P-frames and I-frames to wait until B-frames, sent directly to the *Video Presentation Buffer* arrive. After this, the I-frame and B-frames are also sent to the *Video Presentation Buffer*. See Fig. 3.11 for a visual representation of I, B and P frames.

In Fig. 3.7 the STD for MP2T is shown and in Table 3.9 the meaning of buffers and data

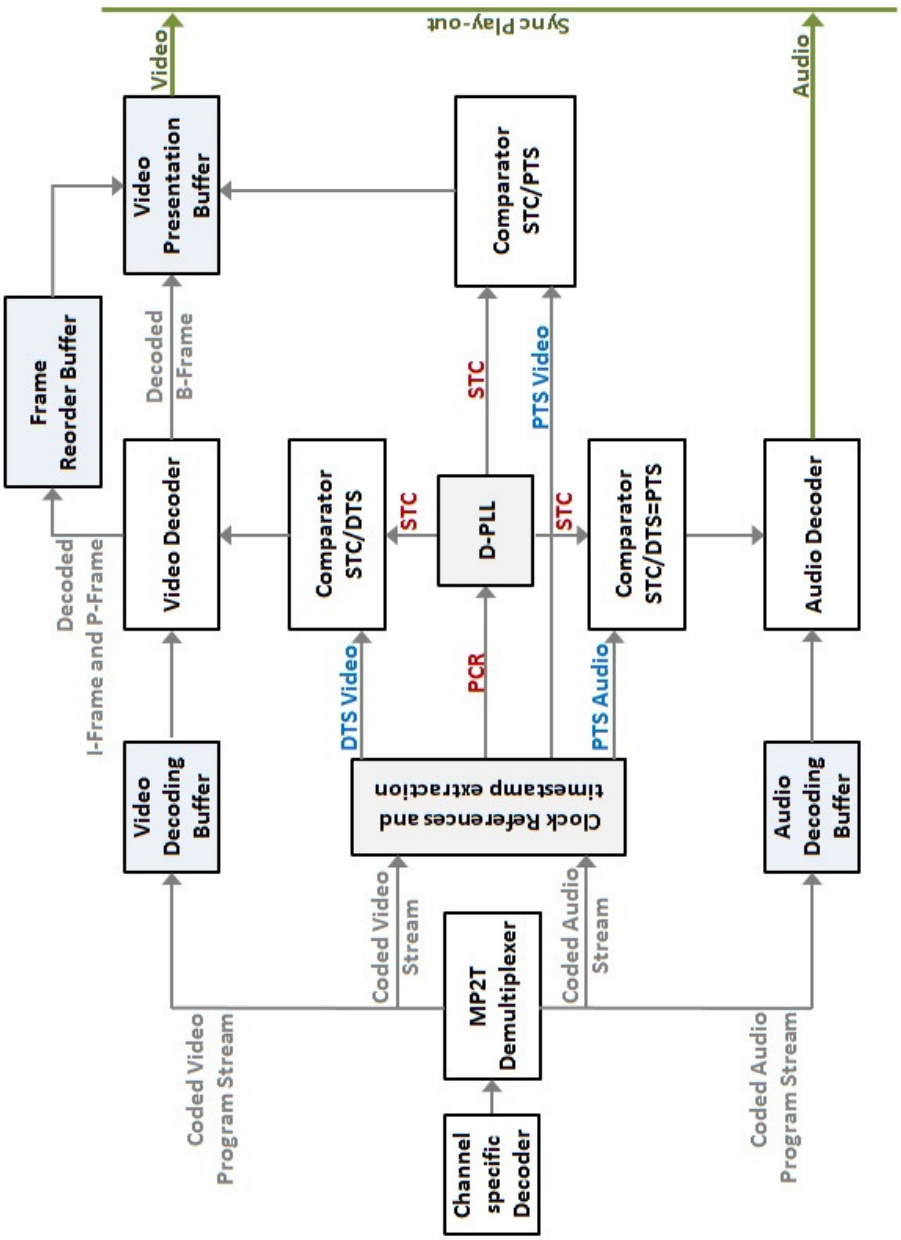


Figure 3.6: Modified diagram from Figure 5.1 in [34]. A diagram on video decoding by using DTS and PTS

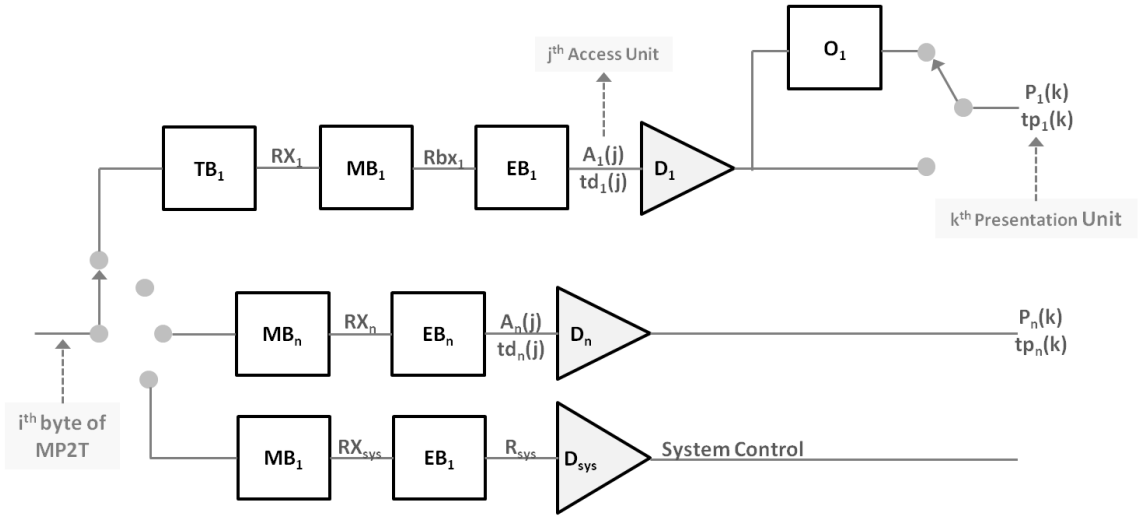


Figure 3.7: Transport Stream System Target Decoder. Figure 2-1 in [30]. Notation is found Table 3.9

in T-STD is listed. The figure shows the three different ES types, video, audio and systems. The top buffer line is an example for video, the middle one for audio, and the bottom one for systems.

3.6.2 Clock References

Clock references are used to introduce timing into an MP2T stream. This section describes firstly how the encoder inserts the clock references within the MP2T stream, secondly how it is transmitted and finally how the decoder uses them to reproduce the encoder's clock at the receiver.

3.6.2.1 Clock References within MP2T Streams

MP2T timing system uses clock references to reproduce the encoder's system clock at the decoder. Within one MP2T stream, multiple programs can be multiplexed, and each with has its own clock reference. To summarize, there are three clock references within the MP2T streams, Program Clock References (PCR), Original Program Clock References (OPCR), and Elementary Stream Clock Reference (ESCR). PCR and OPCR are located in the Adaptation Field whereas ESCR within the ES header. The packetisation process from ES to PES and finally MP2T is described in Fig. 3.8a. The main time fields related are drawn in Fig. 3.8b and the PES fields in 3.8c. Usually one PES is conveyed in multiple MP2T packets

The clock system used at decoder is the System Clock Frequency (SCF). SCF in MP2T

Variable	Meaning
i, i', i"	Byte index in the MP2T. First byte is zero
j	Index of AUs in the ES
k, k', k"	Presentation units index ES
n	ES index
p	MP2T index packet
t(i)	Arrival time in seconds of i^{th} byte of the MP2T
PCR(i)	Value PCR
$\mathbf{A_n(j)}$	j^{th} AU in the n^{th} ES
$\mathbf{td_n(j)}$	Decoding Time (s) of the j^{th} access unit
$\mathbf{P_n(k)}$	k^{th} presentation unit
$\mathbf{tp_n(k)}$	Presentation Time (s) of the k^{th} presentation unit
t	Time in second
$\mathbf{F_n(t)}$	Fullness (bytes) on the STD for n^{th} ES at time t
$\mathbf{B_n}$	ES n^{th} main buffer. Only present in audio ES
$\mathbf{BS_n}$	Size (bytes) in B_n
$\mathbf{B_{sys}}$	Main buffer for system information within the STD
$\mathbf{BS_{sys}}$	Size (bytes) in BS_{sys}
$\mathbf{MB_n}$	n^{th} ES Multiplexing buffer. Only present in video ES
$\mathbf{MBS_n}$	Size (bytes) in MBS_n
$\mathbf{EB_n}$	n^{th} ES buffer. Only present in video ES
$\mathbf{EBS_n}$	Size (bytes) in EBS_n
$\mathbf{TB_{sys}}$	Transport buffer for system information
$\mathbf{TBS_{sys}}$	Size (bytes) in TBS_{sys}
$\mathbf{TB_n}$	Transport buffer for n^{th} ES
$\mathbf{TBS_n}$	Size (bytes) in TBS_n
$\mathbf{D_{sys}}$	System Information decoder for PS n^{th}
$\mathbf{D_n}$	n^{th} ES decoder
$\mathbf{O_n}$	n^{th} ES re-order buffer
$\mathbf{R_{sys}}$	Rate B_{sys} data is removed
$\mathbf{Rx_n}$	Rate TB_n data is removed
$\mathbf{Rbx_n}$	Rate MB_n data is removed for the leak method
$\mathbf{Rbx_n(j)}$	Rate MB_n data is removed for vbv_delay
$\mathbf{Rx_{sys}}$	Rate TB_{sys} data is removed
$\mathbf{R_{es}}$	Video ES rate

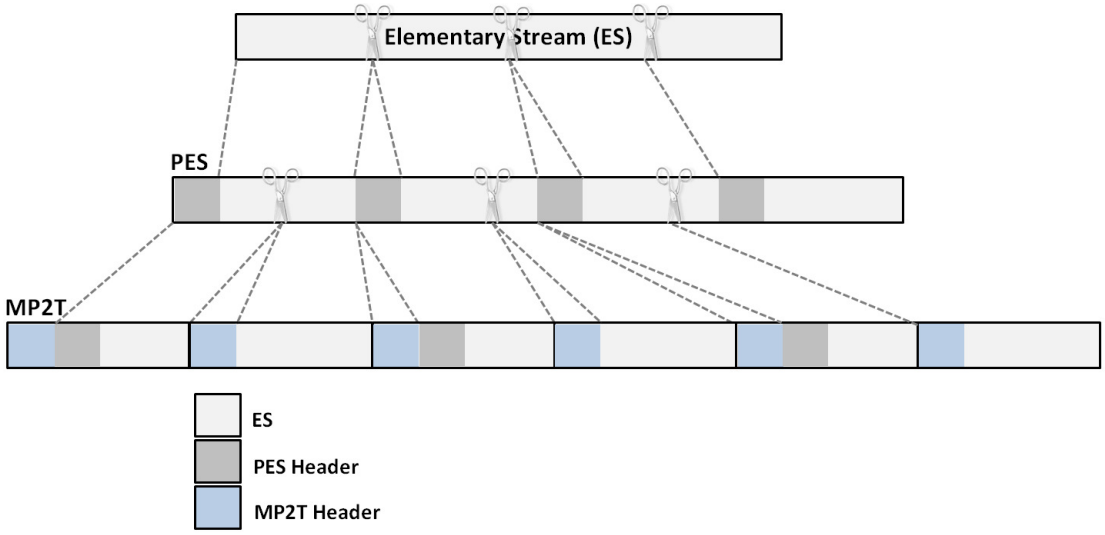
Table 3.9: Notation of variables in the MP2T T-STD [30] for Fig. 3.7

T-STD is always at 27MHz, and must satisfy the following requirements [30]:

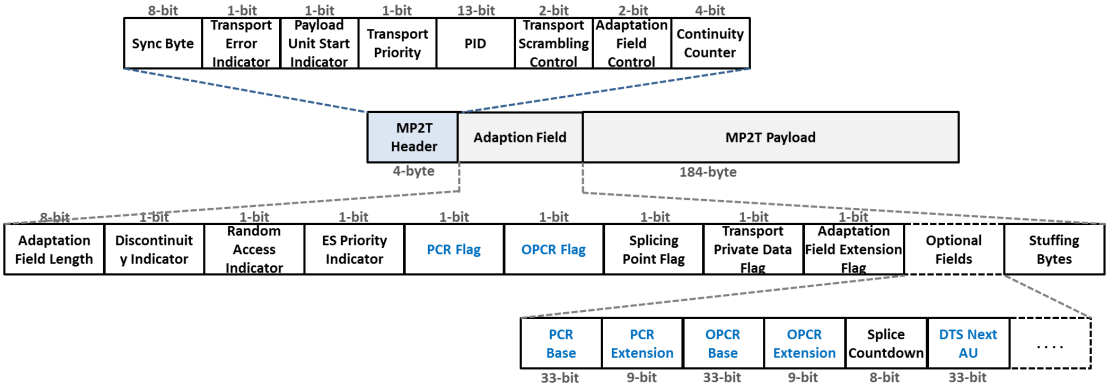
$$27MHz - 810Hz \leqslant SCF \leqslant 27MHz + 810Hz \tag{3.1}$$

$$SCFChangeRate \leqslant 75 \cdot 10^{-3} Hz/s \tag{3.2}$$

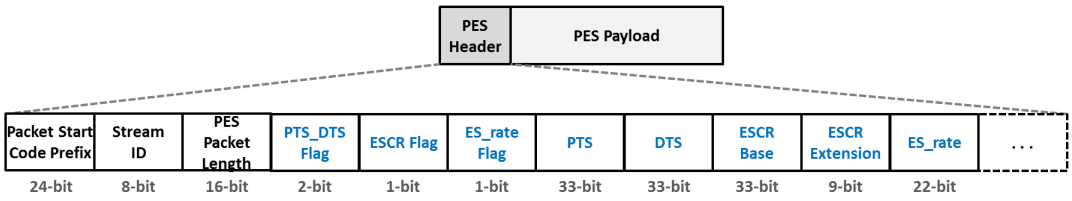
The most important and compulsory field is the PCR. PCR is a 27MHz frequency clock conveyed in 42 bits between two different fields PCR_{base} (33-bit) and PCR_{ext} (9-bit). *PCRflag*



(a) ES, PES, MP2T process



(b) MP2T packet structure



(c) PES packet structure

Figure 3.8: MP2T and PES packet structure

signals its presence.

The 44-bit PCR values can be calculated from the two PCR fields, PCR_{base} and PCR_{ext}. The following equations from are applied [30]:

$$PCR(i) = PCR_{base}(i) \cdot 300 + PCR_{ext}(i) \tag{3.3}$$

$$PCR_{base} = \left(\frac{SCF \cdot t(i)}{300}\right) \%2^{33} \tag{3.4}$$

$$PCR_{ext} = \left(\frac{SCF \cdot t(i)}{1}\right) \%300 \tag{3.5}$$

The parameter i is the byte index of the last PCR_{base} bit. The parameter $t(i)$ is the time when i^{th} byte arrives at T-STD.

The transport rate (TR) of PCR values is calculated using the following equation [30]:

$$TR(i) = \frac{(i' - i'') \cdot 27MHz}{PCR(i') - PCR(i'')} \tag{3.6}$$

The arrival time of byte i^{th} at the T-STD is based on the PCR, SCF and the TR. using the following equation [30]:

$$t\,(i) = \frac{PCR\left(i''\right)}{SCF} + \frac{i - i''}{TR\,(i)} \tag{3.7}$$

The other clock reference OPCR follows exactly the same structure and frequency as PCR. It also has an $OPCR_{base}$ and $OPCR_{ext}$ but this clock reference is used to reconstruct an MP2T stream from the original stream. Its presence is signalled by the *OPCRflag*.

The last Clock Reference is the ESCR, located at the PES Header. The flag *ESCRflag* signals its presence. It is used when the PES packets are not packetised within the MP2T stream. Thus, the clock references need to be conveyed within the PES. The structure and frequency is identical to PCR or OPCR, two fields, $ESCR_{base}$ (33-bit), and $ESCR_{ext}$ (9-bit). Appendix C summarises all MPEG Clock References.

Finally, the last method to transmit timing information about the clock references is the System Clock Descriptor (SCD). The descriptor has the fields listed in Table 3.10. SCD is the means to inform the decoder of the Clock Accuracy (CA) values. CA is 30ppm (parts per million) unless the field CA_{int} is different from zero. CA frequency ($\text{CA}_{\text{frequency}}$) is calculated using CA_{int} and CA_{exp} in the equation [30]:

$$CA_{frequency} = \begin{cases} 30ppm & \text{if } CA_{\text{int}}{=}0 \\ CA_{\text{int}} \cdot 10^{-\text{CAexp}} & \text{if } CA_{\text{int}}{\neq}0 \end{cases} \tag{3.8}$$

where parameter CA_{int} is the Clock Accuracy Integer and parameter CA_{exp} is the Clock Accuracy Exponent.

3.6.2.2 Encoder and decoder sync

The clock reference are inserted at the MP2T stream at a 27MHz frequency, named PCR. The decoder's clock system has its own clock system called System Time Clock (STC) running at

Field	Bits	Description	Utility
Descriptor_tag	8	value 11 for MP2P and MP2T	It signals a System Clock Descriptor
Descriptor_length	8	Descriptor bytes size after the descriptor_length field	To know the end of descriptor
External_clock reference_indicator	1	Flag that indicates the reference to an external clock	It references an external clock accuracy
Reserved	1		
Clock_accuracy_ integer	6	Integer of frequency accuracy of system clock (ppm)	Integer is used to calculate clock accuracy if it is higher than 30ppm
Clock_accuracy_ exponent	3	Exponent of frequency accuracy of system clock (ppm)	Exponent is used to calculate clock accuracy if it is higher than 30ppm
Reserved	5		

Table 3.10: System Clock Descriptor Fields and Description [30]

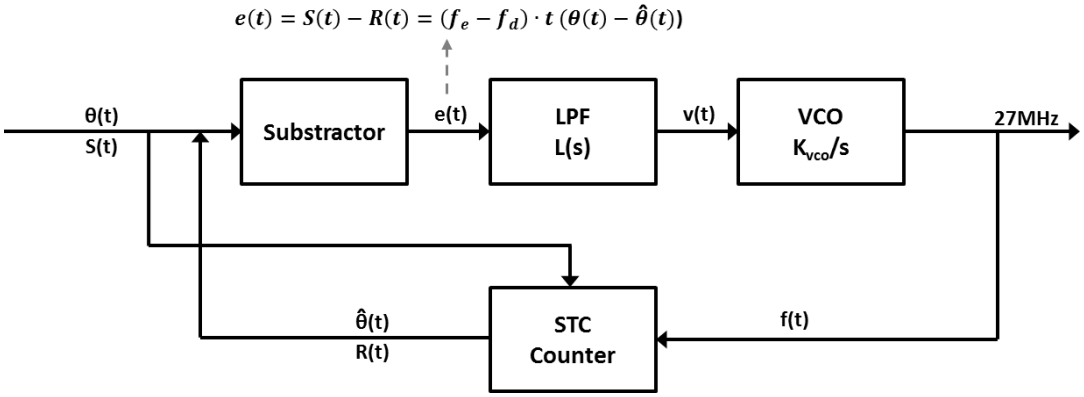


Figure 3.9: A model for the PLL in Laplace-transport domain modified. Figure 4.5 in [34]

approximately the same frequency.

To sync the STC on decoder to the encoder's PCR, MP2T streams use a Phase Lock-Loop (PLL). A model of the PLL is described in Fig. 3.9 which receives the encoder's PCR values and syncs the STC frequency at the decoder to this.

In Fig. 3.10 the actual PCR function can be seen. The incoming PCRs, although arriving at discrete points in time, is presume to emulate a continuous-time function:

$$S(t) = f_e \cdot t + \theta(t) \tag{3.9}$$

f_e is the encoder's system clock, and $\theta(t)$ is *'the incoming clock's phase relative to a designated time origin'* [34].

'The actual incoming clock signal $\widehat{S}(t)$ is a function with discontinuities at the time instants at which PCR values are received, with slope equal to f_d for each of its segments, where f_d is

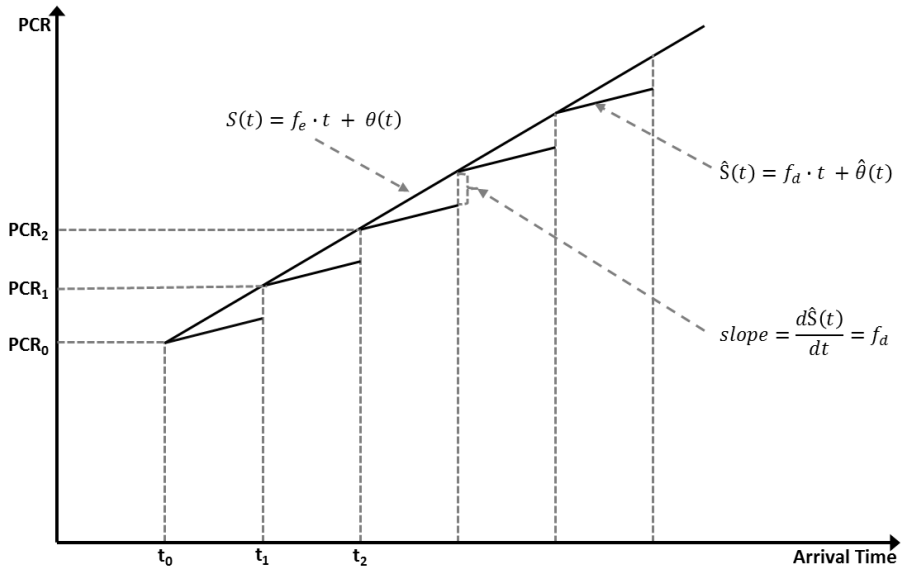


Figure 3.10: Actual PCR and PCR function used in analysis. Figure 2 in [85]

the running frequency of the decoder's clock' [34].

$\widehat{\theta}(t)$ is the decoder's clock function with discontinuities at time instants in time running at f_d frequency.

$$\widehat{S}(t) = f_d \cdot t + \widehat{\theta}(t) \tag{3.10}$$

The time increment of PCR arrivals is not greater than 0.1s following the MPEG-2 Standard. Therefore, it guarantees that the two functions, $\theta(t)$ and $\widehat{\theta}(t)$, are very close. That is why $\theta(t)$ is used instead of $\widehat{\theta}(t)$ [34].

$$slope = \frac{\mathrm{d}S(t)}{\mathrm{d}t} = f_d \tag{3.11}$$

Once $S(t)$ or $\theta(t)$ arrives to the PLL decoder the subs-tractor compares with $R(t)$ or $\widehat{\theta}(t)$ to generate $e(t)$:

$$e(t) = S(t) - R(t) = (f_e - f_d) \cdot t(\theta(t) - \widehat{\theta}(t)) \tag{3.12}$$

Taking into account that if $f_e = f_d$ then $e(t) = \theta(t) - \widehat{\theta}(t)$. Based on function $e(t)$ LPF calculates the values of $v(t)$.

$$\frac{\mathrm{d}\widehat{\theta}(t)}{\mathrm{d}t} = K_{VCO} \cdot v(t) \tag{3.13}$$

The VCO, based on this input, generates the $f(t)$ which will be the new frequency to feed the STC Counter. The process is locked while $\theta(t) = \widehat{\theta}(t)$.

In the particular case of MPEG-2 Systems PLL the system' aims to achieve encoder's and decoder's sync, therefore PLL will be locked until $f_e = f_d$, which is 27MHz frequency.

In MP2T streams there is a constant relationship between both the audio sampling rate

Audio Sampling Frequency (kHz)	**16**	**32**	**22.05**	**44.1**	**24**	**48**
SCASR	27k/16	27k/32	27k/22.05	27k/44.1	27k/24	27k/48

Table 3.11: SCAR Table from [30]

Frame Rate (kHz)	**23.976**	**24**	**25**	**29.97**	**30**	**50**	**59.94**	**60**
SCFR	1126125	1125000	1080000	900900	900000	540000	450450	450000

Table 3.12: SCFR Table from [30]

and frame rate and the System Clock Frequency (SCF), 27MHz. The former is the System Clock Audio Sampling Rate (SCASR) and the latter is the System Clock Frame Rate (SCFR). This relationship is established by the equations [30]:

$$SCASR = \frac{SCF}{audio_sample_rate\ in\ T-STD} \tag{3.14}$$

$$SCFR = \frac{SCF}{frame_rate\ in\ T-STD} \tag{3.15}$$

In Table 3.11 all possible values for SCASR can be found and in Table 3.12 all possible values for SCFR can be found.

3.6.3 Timestamps

There are two types of timestamp Decoding (DTS) and Presentation Timestamps (PTS). These timestamps outline a discrete moment in time when an AU shall be decoded or presented. The purpose of two different timestamps is based on the fact that for video AU shall, in some cases, be decoded prior to be presented. Appendix C contains the Table 10 that summarises all MPEG timestamps.

In audio AUs, PTS is always equal to DTS. Therefore, instant audio decoding is presupposed.

In video the PTS and DTS values are based on the presence of I, P and B-frames. I-frames are self contained and thus decoded within their frame, P-frames are decoded using information from a previous frame and finally, B-frames use information from a previous frame and a posterior frame. Fig. 3.11 illustrates a distribution of a Group of Pictures (GOP) where I, P and B-frames can be found as well as the dependencies between frames. A real example from a video stream can be seen in Fig. 3.12, where PCR and PTS values shown are real (DTS values are only for demonstration purposes).

Following Fig. 3.11 it can be seen that P-frame$_4$ relies on I-frame$_1$ therefore, I-frame$_1$ needs to be previously decoded. However B-frame$_2$ and B-frame$_3$ rely on I-frame$_1$ and P-frame$_4$.

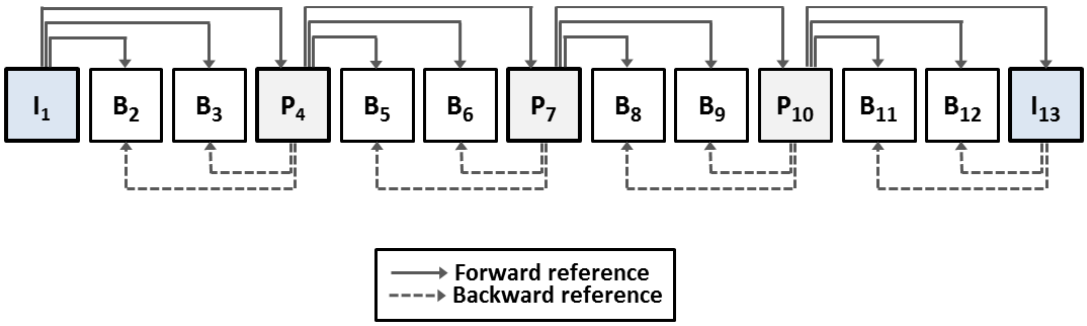


Figure 3.11: A GOP high level distribution

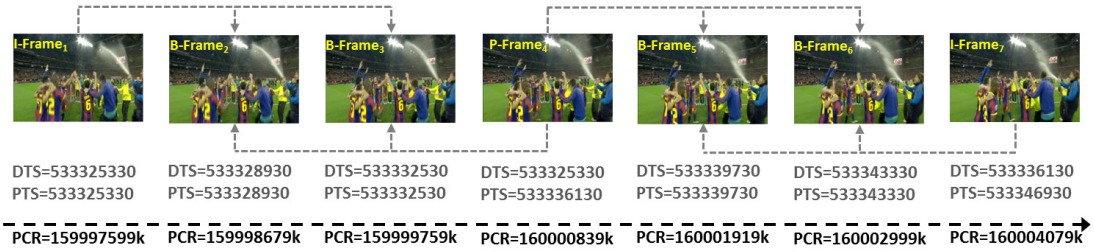


Figure 3.12: A GOP High Level distribution with MP2T timestamps (DTS and PTS) and clock references (PCR)

If the MPEG-2 video stream does not have B-frames then timestamps follow the audio pattern whereby DTS equals PTS because when a P-frames arrives there is always the guarantee that the previous frames have been already been decoded. An absence of B-frames means pictures reach the decoder's buffer at presentation time.

The presentation order is not maintained at decoder's buffer if B-frames are present in MP2T stream. When B-frames are present in video, then DTS is different to PTS values, thus, some frames arriving after the B-frames should be decoded before the presentation time so that the frame is available for prior B-frame to be decoded.

B-frames always have PTS equal to DTS, thus, only PTS is coded within the MP2T stream. DTS and PTS of I and P frames vary in a time difference which is always a *'multiple of the nominal picture period'* [34] [84].

The timestamping process requires information, based on which timestamp values are set [34] [84]:

- Picture Type: I, P and B-frame
- Temporal Reference: Count of pictures in presentation order (10-bit)
- Picture Encoding Timestamp (PETS): A PCR fraction time value which was locked by picture sync (33-bit)

Configuration	Mode		Film Mode
Configuration 1	B-frame disable	mode=0	No Film Mode
Configuration 2	B-frame disable	mode=0	Film Mode
Configuration 3	Single B-frame	mode=1	No Film Mode
Configuration 4	Single B-frame	mode=1	Film Mode
Configuration 5	Double B-frame	mode=2	No Film Mode
Configuration 6	Double B-frame	mode=2	Film Mode

Table 3.13: Configuration Timestamping [84]

Pictures	Transmitted Fields	Repeat First Field Flag	Top Field First Flag	Displayed Field
frame A	A1 → A2	1	1	A1 → A2→ A1
frame B	B1 → B2	0	0	B2 → B1
frame C	C1 → C2	1	0	C2 → C1 → C2
frame D	D1 → D2	0	1	D1 → D2

Table 3.14: Film Modes States from Table 6.2 in [84]

- Film mode [1]

The timestamping process of a picture depends on this information, including the picture mode. There are three video coding modes which classify the GOP structures [34]:

- Mode 1: No B-frames present
- Mode 2: One B-frame between each I or P-frame
- Mode 3: Two B-frames between each I or P-frame

The list of all possible timestamping configuration is found in Table 3.13 and the possible Film Modes states in Table 3.14.

The calculation of DTS_i is based on the PETS and T_d which is *'the nominal delay from the output of the encoder to the output of the decoder'* [84].

$$DTS_i = PETS_i + T_d \tag{3.16}$$

The time difference F between PTS and DTS is equal to the nominal picture time in no film mode. This time difference F is the one used in every configuration in the timestamp process. For NTSC systems the value:

$$F = \frac{90 \cdot 10^3}{29.97} = 3003 \tag{3.17}$$

[1] '*In film mode, two repeated fields have been removed from each ten-field film sequence by the MPEG-2 video encoder*' [84]. In countries such as USA and Canada video is coded at 59.94 fields per second (fps), rounded to 60fps, which is encoded and transmitted at 29.97 Frames per Second (FPS), rounded to 30FPS. Film mode is the mechanism of converting video from 24FPS to 30FPS by adding one repeated video Frame every 4 original video Frames [84]

Video Type	Codec	Film Mode	DTS	PTS	Display Duration
m=1		**No**	$PETS_i+T_d$	DTS_i	F or 1.5F
		Yes	$PETS_i+T_d$	DTS_i	F or 1.5F
m=2		**No**	$PETS_i+T_d$	DTS_i+F	F or 1.5F
			$PETS_i+T_d$	DTS_i+2F	F or 1.5F
		Yes	$PETS_i+T_d$	DTS_i+0.5F	F or 1.5F
			$PETS_i+T_d$	DTS_i+F	F or 1.5F
			$PETS_i+T_d$	DTS_i+2F	F or 1.5F
			$PETS_i+T_d$	DTS_i+2.5F	F or 1.5F
m=3		**No**	$PETS_i+T_d$	DTS_i+F	F or 1.5F
			$PETS_i+T_d$	DTS_i+2F	F or 1.5F
			$PETS_i+T_d$	DTS_i+3F	F or 1.5F
		Yes	$PETS_i+T_d$	DTS_i+F	F or 1.5F
			$PETS_i+T_d$	DTS_i+3F	F or 1.5F
			$PETS_i+T_d$	DTS_i+3.5F	F or 1.5F
			$PETS_i+T_d$	DTS_i+4F	F or 1.5F

Table 3.15: PTS and DTS General Calculation [84]

Bits	Meaning	Description
00	No timestamps present	
01	Value forbidden	
10	PTS present	Presentation equal to decoding time
11	PTS and DTS present	Presentation different from decoding time

Table 3.16: Values of *PTS_DTS_flag* [30]

and for PAL systems the value is:

$$F = \frac{90 \cdot 10^3}{25} = 3600 \tag{3.18}$$

The principles to encode the timestamp are based on each of the possible timestamping configurations listed in Table 3.13. They are also based on the fields *RepeatFirstFieldFlag* and *TopFieldFirstFlag* which determine the Film Mode States listed in Table 3.14.

A brief summary of possible values of PTS and DTS in different video codec mode, Film mode Status, is listed in Table 3.15. In the table, all possible values of PTS are shown without specifying all possible cases and conditions. All detailed rules in each case can be found in multiple tables in [34] [84].

In MP2T both timestamps, DTS and PTS, are 33-bit as shown in Fig. 3.8c located in the PES header at 90KHz resolution. The *PTS_DTS_flag* (2-bit) indicates the presence of both fields.

In Table 3.16 possible flag values are included. In the case of audio or video with no B-frames, as already indicated, DTS equals PTS and *PTS_DTS_flag* value is *10*. In the case of

video with B-frames *PTS_DTS_flag* can have value *10* or *11*.

To obtain the PTS or DTS the following formulae are used which are based on the presentation and decoding time.

$$PTS = \frac{(SCF \cdot (tp_n(j)))}{300} \% 2^{33} \tag{3.19}$$

$$DTS = \frac{(SCF \cdot (td_n(k)))}{300} \% 2^{33} \tag{3.20}$$

where parameter $tp_n(j)$ is the PTS (in seconds) of j^{th} AU within ES_n. and parameter $tp_n(k)$ is the DTS (in seconds) of k^{th} AU within ES_n. ES_n is the n^{th} ES related to DTS.

The last timestamp found in MP2T is *DTS_next_AU*. Its function is to facilitate media splicing. Splicing is the technique used to join the end of one media stream to the beginning of another one. If the flag *seamless_splice_flag* equals zero then splicing type is ordinary. On the contrary, when flag is set, the fields *DTS_next_AU* (33 bits) and *splice_type* (4 bits) are present. The latter indicates the splice decoding delay and the maximum splice rate. *DTS_next_AU* signals the decoding time of the AU found just after the splicing point.

3.6.3.1 Timestamp Errors

The clock-recovery process at the decoder supervises the arriving PCRs within the MP2T stream and corrects them when it is necessary. The decoder's PLL monitors encoder's PCRs and compares them with the decoder's clock system to detect discontinuities.

When a discontinuity is detected the decoder's STC is updated with the new PCR. The picture is then decoded when DTS equals STC. Once the STC has been updated then the PLL returns to monitor the encoder's PCRs values [84].

3.6.4 ETSI TS 102 034: Transport MP2T Based DVB Services over IP Based Networks. MPEG-2 Timing Reconstruction

In ETSI TS 102 034 [8], Annex A describes the MPEG-2 Timing reconstruction based on the usage of RTI defined in standard MPEG-2 part 9 [86].

This standard specifies the MPEG-2 Timing reconstruction based on the relationship between PCR values and RTP timestamps. The equations from 13818-1 [30] to calculate the transport rate (equation 3.21) and arrive time of a byte (equation 3.22) equations are:

$$TR(i) = \frac{(i' - i'') \cdot 27MHz}{PCR(k) - PCR(k-1)} \tag{3.21}$$

, where i is the byte index of the last bit of the next *PCR_base*, where i"< i< i' and k is the first PCR index.

$$t(n+1) = \frac{PCR(k)}{27MHz} - \frac{P}{TR(i)} \tag{3.22}$$

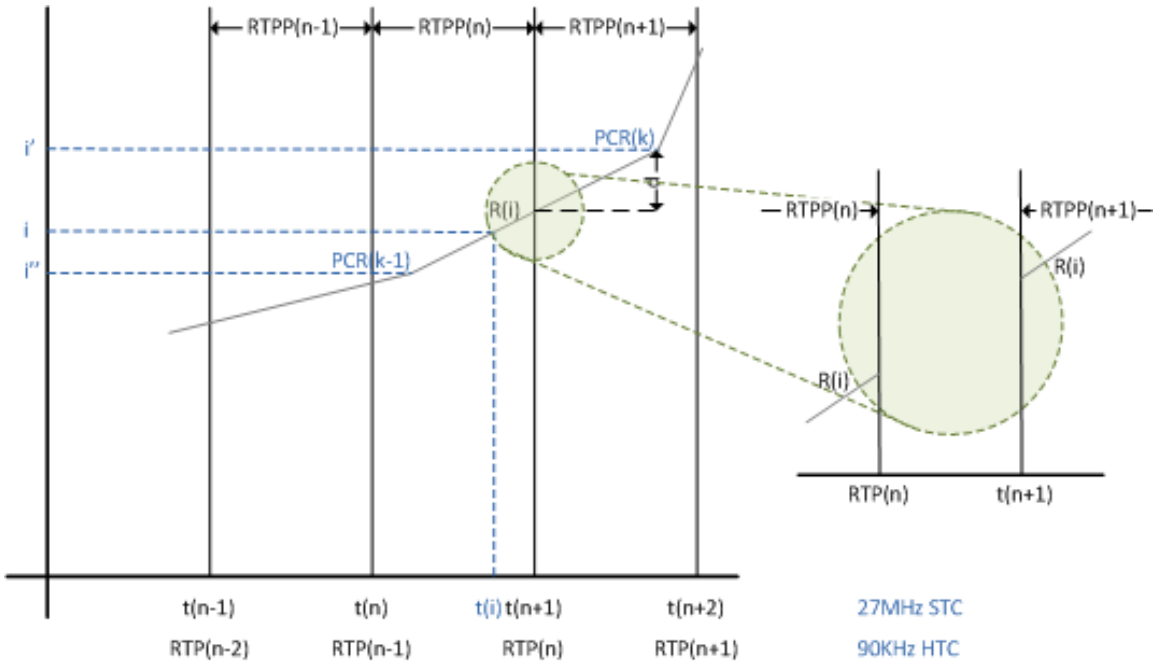


Figure 3.13: Association of PCRs and RTP packets. Fig A.1 in ETSI 102 034 [8]

where i is the i^{th} byte index within the TS, with i"$<$ i. The parameter $i^{'th}$ is the $i^{"th}$ byte index of the last bit of the latest PCR_base. TR(i) is the Transport Rate of i^{th} byte. And finally, the parameter PCR(i) is the time encoded in system clock's units from the PCR base and extension fields.

The relationship between PCR and RTP timestamps is established in the following equation, shown in Fig. 3.13. The formula is based on the MP2T transport rate between two consecutive MP2T packets containing PCR values.

$$PCR(k) \cong RTP(n) + 90KHz \cdot \frac{(P+1)}{TR(i)} \tag{3.23}$$

where n is the RTP index, P is the quantity of bytes from the preceding PCR(k) (from equation 3.21) and finally, $TR(i)$ is the transport rate calculated in equation 3.21 [30].

This formula states the relationship between a PCR (27MHz frequency) and the RTP timestamp in the RTP packet header conveying the MP2T with the PCR value.

The problem with this relationship is that it assumes that two consecutive RTP packets convey MP2T packets containing PCRs values. This is not feasible because it is recommended up to seven MP2T packets can be carried within a RTP packet therefore, this condition is hardly ever met [87]. For example, analysis of a real MP2T file has yielded a total of 3993 PCR values with results summarised in Table 3.17. PCR_{i+1} and PCR_i are two consecutive PCR values where j is the number of MP2T packets between them in equation 3.24. It was

	Δ PCRs	Occurrences	%	SubTotal
PCR_{i+1}-PCR_i<=7	PCR_{i+1}-PCR_i=0	0	0	34 (0.85%)
	PCR_{i+1}-PCR_i=1	5	0.12	
	PCR_{i+1}-PCR_i=2	6	0.15	
	PCR_{i+1}-PCR_i=3	5	0.12	
	PCR_{i+1}-PCR_i=4	7	0.17	
	PCR_{i+1}-PCR_i=5	1	0.02	
	PCR_{i+1}-PCR_i=6	6	0.15	
	PCR_{i+1}-PCR_i=7	4	0.10	
PCR_{i+1}-PCR_i>7	50>PCR_{i+1}-PCR_i>=8	285	7.13	3959 (99.14%)
	100>PCR_{i+1}-PCR_i>=50	2833	70.94	
	150>PCR_{i+1}-PCR_i>=100	826	20.68	
	200>PCR_{i+1}-PCR_i>=150	15	0.37	
TOTAL:		3993	100	

Table 3.17: Analysis of PCR values in a real MP2T sample. Analysis of number of MP2T packets between two consecutive MP2T packets containing PCRs values

found that only in 0.85% of the cases were two consecutive RTP packets have PCRs values. The findings from Table 3.17:

$$j \begin{cases} <= 7 \rightarrow \quad 0.85\% \rightarrow 8 \; MP2T packets \; out \; of \; 3993 \\ > 7 \quad \rightarrow 99.14\% \rightarrow 3958 \; MP2T packets \; out \; of \; 3993 \end{cases} \tag{3.24}$$

3.7 MPEG-4 Timelines

In this section, the scope of synchronisation in MPEG is extended beyond the scope of the prototype to include MPEG-4. In particular, it examines the two timing systems used in MPEG-4, MPEG-4 Sync Layer (MPEG-4 SL) and M4Mux[1] are described. M4Mux is a low overhead and low delay Sync Layer tool to provide interleaving and instant bitrate SL streams.

The clock references and timestamps in MPEG-4 are conveyed in the MPEG-4 SL header. The added features in this system are based on information conveyed in descriptors such as *SL Config*, *Decoder Config*, *ES* and *M4Mux Timing* Descriptor. The following sections describe how the information is organised within the MPEG-4 descriptors and within the MPEG-4 SL Header.

3.7.1 STD

The Delivery Multimedia Integration Framework (DMIF) Application Interface (DAI) receives the streamed data as shown above in Fig. 3.14. The demultiplexer transmits the correspondent stream to its decoding system. The Access Units (AU) wait within the decoding buffer until

[1]FlexMux and M4Mux. FlexMux is used in MPEG-2 part 1 and M4Mux in MPEG-4 part 1. In document ISO/IEC JTC 1/SC 29/WG 11 N5677 explains that FlexMux is a copyrighted term therefore M4Mux should be used

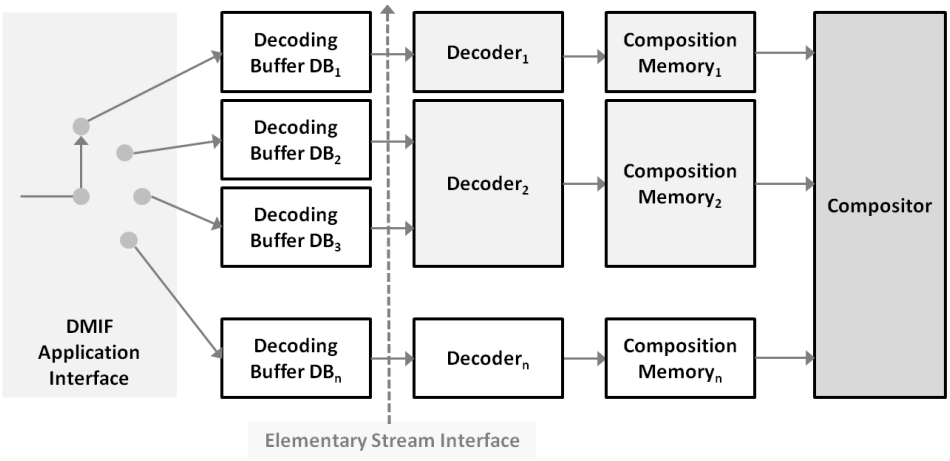


Figure 3.14: System Decoder's Model for MPEG-4. Figure 2 in [33]

DTS notifies them to be extracted from the buffer and sent to the decoder. AUs are decoded and transformed into Composition Units (CU) by the decoder, then CUs are sent by the decoder to the composition buffer waiting until indicated by CTS to be transferred to the Compositor Unit where all units from different streams are arranged for further media stream play-out [33].

The System Decoder Model provides the demultiplexing tools to access data streams (DAI), the decoding buffer system for each type of the elementary stream, elementary stream decoders, the composition buffer systems for every decoder type, and finally, the compositor prior the media stream presentation [33]

3.7.2 Clock References

The encoder's MPEG-4 Object Time Base (OTB) is reproduced at decoder via the Object Clock References (OCR). In MPEG-4 SL the clock references are conveyed within the Header. The flag *OCR_flag* indicates the presence of the *OCR* field. This field information is conveyed within the *SL Config* Descriptor. Field *OCRlength* (8-bit) indicates the OCR number of bits and *OCRresolution* (32-bit) the OCR resolution. The structure is highlighted in Fig. 3.15.

The Object Clock Reference (OCR) is used to carry the OTB in the elementary streams from decoder to the terminal's decoder. OCR's value is established as *'the value of the OTB at the time the sending terminal generates the object clock reference timestamp'* [33] and it is conveyed within the SL packet header of an SL-packetised stream. The moment the receiver should evaluate the OCR is specified as when *'its last bit is extracted at the input of the decoding buffer'* [33]. The main differences between the Clock References are outlined in Table 10 in Appendix C. The location of OCR and OTB clock references is shown in Fig. 3.16 but the list with the main differences between them is found in Table 3.18 [33].

The time in seconds of the OCR values can be extracted using the SL Config Descriptor

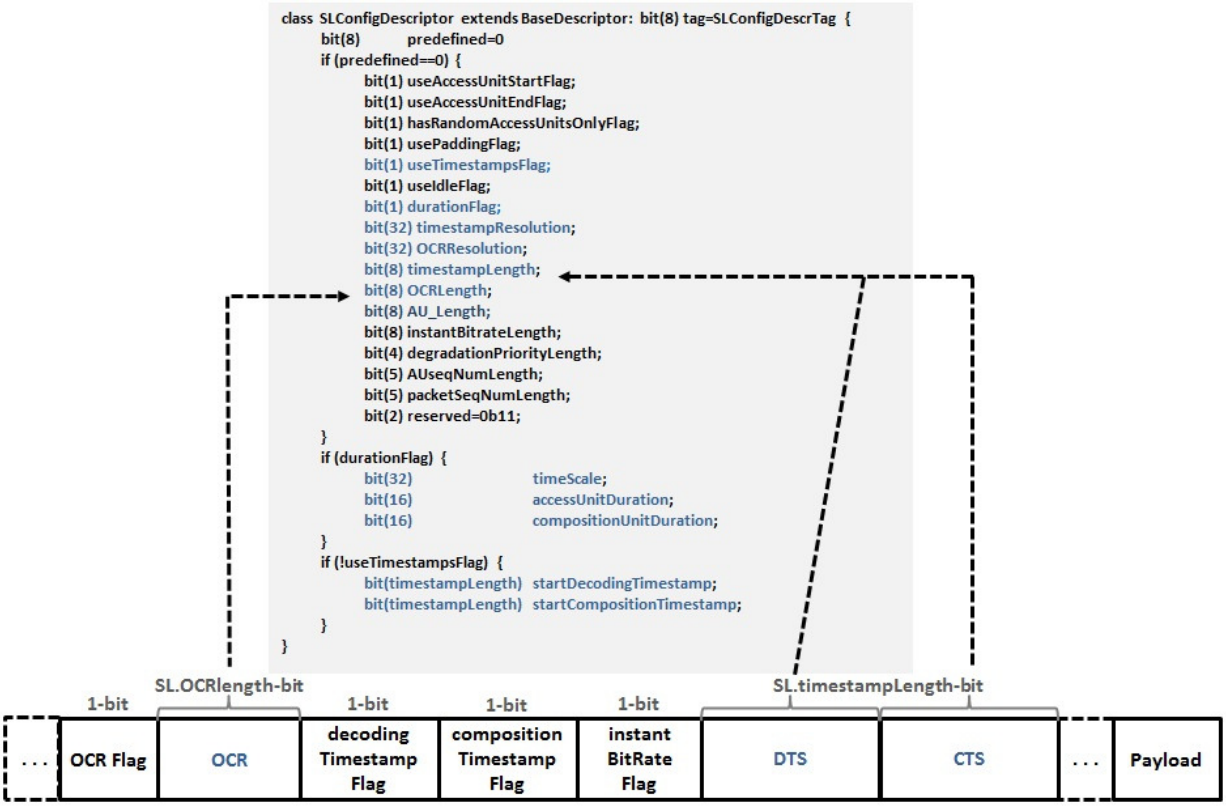


Figure 3.15: MPEG-4 SL Descriptor. Time Related fields

OTB	STB
Data Stream notion of time	Terminal notion of time
Resolution is defined by the application or the profile	Resolution is implementation dependent
Timestamps in the data stream relate to the OTB	Terminal actions relate to the STB
OTB is sent to the terminal through the OCR	

Table 3.18: Comparison between OTR and OCR clock references

fields using the following equation [33]:

$$t_{OCR} = \left(\frac{OCR}{OCR_{res}} + k \cdot \frac{2^{OCR_{len}}}{OCR_{res}} \right) \tag{3.25}$$

OCR values can be ambiguous therefore a parameter k is introduced to indicate the number of wrap-arounds. Every time a clock reference is received, to prevent equivocal values, the following condition shall be meet [33]. The value k which should be the one that minimizes:

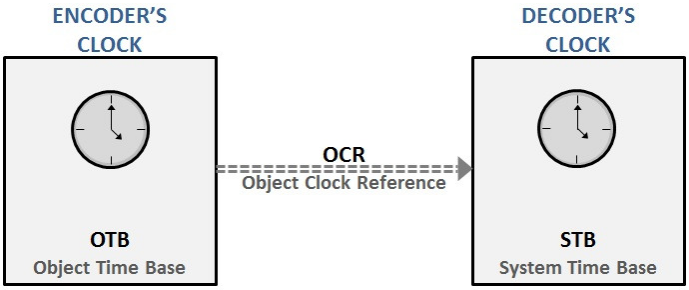


Figure 3.16: MPEG-4 Clock References location

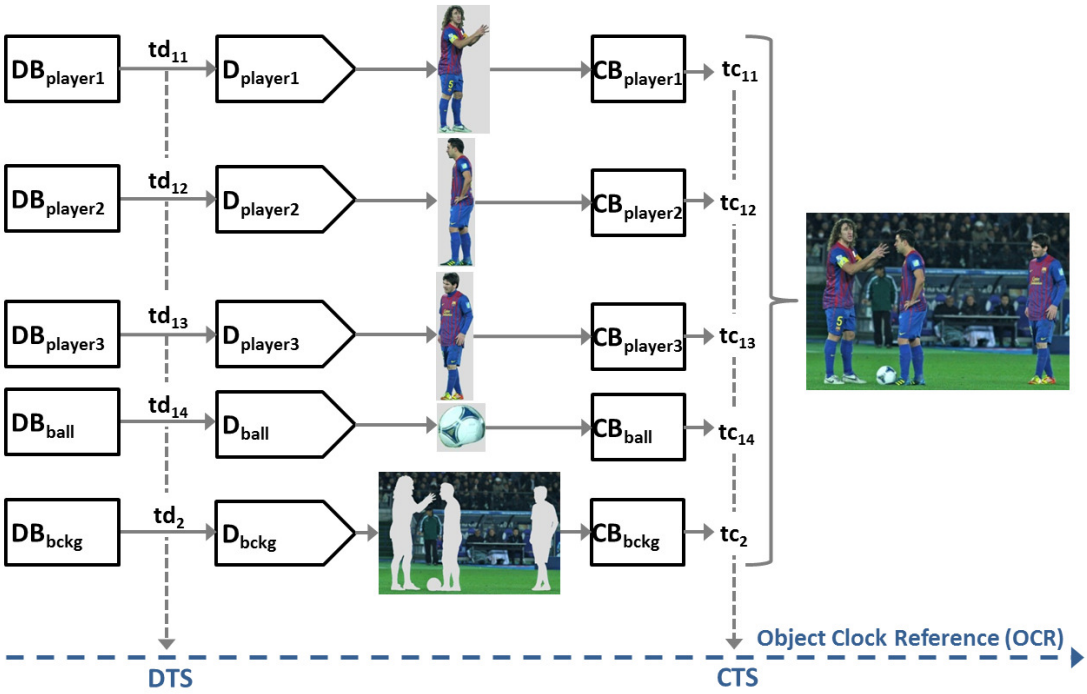


Figure 3.17: VO in MPEG-4 and the relationship with timestamps (DTS and CTS) and clock references (OCR)

$$|t_{OTBestimated} - t_{ts}(k)| \tag{3.26}$$

Fig. 3.17 provides an example of MPEG-4 visual objects within a picture and the difference when the timestamps, DTS and CTS, are synced with the clock references OCRs. All objects are decoded at DTS time to be composed at CTS time which is the presentation time.

Fig. 3.17 also illustrates the principles of DTS and CTS related to Video Objects (VO) are depicted. The AUs are waiting in the Decoding Buffers ($DB_{player1}$, $DB_{player2}$, $DB_{player3}$, DB_{ball}

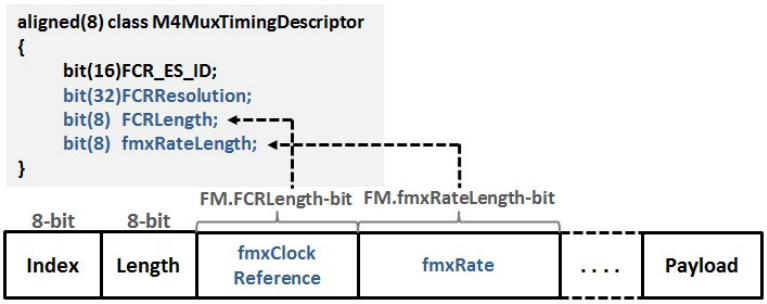


Figure 3.18: M4Mux Descriptor

and DB_{bckg}). VOs are decoded at DTS time[1] (td_{11}, td_{12}, td_{13}, td_{14} and td_2) (football players, the ball and background), and, once objects are decoded, the CUs wait in the composition buffer ($CB_{player1}$, $CB_{player2}$, $CB_{player3}$, CB_{ball} and CB_{bckg}) until the composition time (tc_{11}, tc_{12}, tc_{13}, tc_{14} and tc_2). A picture is composed from all the VOs at CTS time. In the figure the objects are displayed after being decoded at the DTS time instant. Then, at CTS instant, all objects are composed generating the complete frame. Both timestamp instants DTS and CTS are related to the OCR clock reference timeline showed at the bottom of the picture.

There are two descriptors conveying time information, the *ES* Descriptor, in the MPEG-4 SL, and the *M4MuxTiming* Descriptor, within an M4Mux Stream. The ES Descriptor conveys the information *OCR_ES_id* which links the timelines system to an external time base.

The M4Mux has its own clock reference conveyed within the M4Mux Header within the field *fmxClockReference* with a variable number of bits. The clock rate is conveyed within *fmxRate* also with a variable number of bits. The bit size of both fields is indicated in the *M4MuxTiming* Descriptor.

The number of bits are indicated in field *FCRLength* (32-bit) for the *fmxClockReference* and in *fmxRateLength* for the *fmxRate* field. Finally, the FCR resolution will be located within the *M4MuxTiming* Descriptor within the *FCRResolution* field. The M4Mux timing system is highlighted in Fig. 3.18.

The FCR arrival time can be obtained using the following equation [33]:

$$t(i) = \left(\frac{FCR\,(i'')}{FCRres}\right) + \left(\frac{i - i''}{fmxRate(i)}\right) \tag{3.27}$$

3.7.2.1 Mapping Timestamps to the STB

$$t_{SCT} = \frac{\Delta t_{STB}}{\Delta t_{OTB}} \cdot t_{OCT} - \frac{\Delta t_{STB}}{\Delta t_{OTB}} \cdot t_{OTB-START} + t_{STB-START} \tag{3.28}$$

where t_{STB} is the *'composition time of a CU measured in units of t_{STB}. t_{STB} is the current time in the receiving terminal's STB'*. t_{OCT} is the *'composition time of a CU measured in*

[1]For simplicity reasons decoding time td_{11} equals td_{12}, td_{13}, td_{14} and td_2

units of t_{OTB}'. t_{OTB} is *'the current time in the data stream's OTB, conveyed by an OCR'*. $t_{STB\text{-}START}$ is *'value of receiving terminal's STB when the first byte of the OCR timestamp of the data stream is encountered'* [33].

$$\Delta t_{OTB} = t_{OTB} - t_{OTB-START} \tag{3.29}$$

$$\Delta t_{STB} = t_{STB} - t_{STB-START} \tag{3.30}$$

Adjusting the STB to an OTB:

$$t_{STB-START} = t_{OTB-START} \tag{3.31}$$

$$\Delta t_{STB} = \Delta t_{OTB} \tag{3.32}$$

$$t_{SCT} = t_{OCT} \tag{3.33}$$

3.7.2.2 Clock Reference Stream

MPEG-4 SL also uses a single SL stream to exclusively provide clock references so multiple media streams which can share the same timing system. This is done via an MPEG-4 SL which is not conveying media data but only conveying OCRs. This type of stream is called ClockReference Stream. ClockReference stream, as any other, is based on information provided by different Descriptors. The values of the fields within these Descriptors are listed in Table 3.19.

To link one MPEG-4 SL to an external timebase from another ES stream, the fields in the ES Descriptor *OCRstreamFlag* and *OCR_ES_id* (16-bit) are used. The flag indicates this external time base link and the *OCR_ES_id* indicates the ES's id containing the timebase to be applied.

3.7.3 Timestamps

Timestamps in MPEG-4 are slightly different from MP2T streams. DTS is also present although the Composition Timestamp (CTS) is used instead of PTS. PTS in MP2T denotes the presentation timestamp whereas CTS indicates the composition time, the time to compose a CU, which can be composed from multiple AUs.

The presence of DTS and CTS fields is signalled by *decodingTimestampFlag* and *composingTimestampFlag* respectively. Both fields DTS and CTS have *timeStampLength* within the *SL Config* Descriptor. Their resolution is also indicated by field *timestampResolution* also within the *SL Config* Descriptor. Fig. 3.15 shows the fields within an MPEG-4 structure.

Descriptor	Field	Value
SL Packet	It shall no convey a SL packet payload SL packet only conveys OCR values: *OCRResolution* and *OCR-Length*	
Decoder Config	hasRandomAccessUnitsOnlyFlag	1
	objectTypeIndication	0xFF
	bufferSizeDB	0
SL Config	useAccessUnitStartFlag	0
	useAccessUnitEndFlag	0
	useRandomAccessPointFlag	0
	usePaddingFlag	0
	useTimestampsFlag	0
	useIdleFlag	0
	durationFlag	0
	timestampResolution	0
	timestampLength	0
	AULength	0
	degradationPriorityLength	0
	AUseqNumLength	0

Table 3.19: Configuration values from SL packet, *DecoderConfig* Descriptor and *SLConfig* Descriptor when timing is conveyed through a Clock Reference Stream [33]

Two fields within the *SL Config* Descriptor, *timescale* and *AccessUnitduration* are used to obtain the $\mathrm{AU}_{\mathrm{time}}$ and $\mathrm{CU}_{\mathrm{time}}$. The equations are as follows [33]:

$$AU_{time} = AUDuration \cdot \left(\frac{1}{timeScale} \right) \tag{3.34}$$

$$CU_{time} = CUDuration \cdot \left(\frac{1}{timeScale} \right) \tag{3.35}$$

The time instant related to DTS and CTS values are calculated via the following equations [33]:

$$t_{DTS} = \left(\frac{DTS}{SL.TS_{res}} + k \cdot \frac{2^{TS_{Len}}}{TS_{res}} \right) \tag{3.36}$$

$$t_{CTS} = \left(\frac{CTS}{SL.TS_{res}} + k \cdot \frac{2^{TS_{Len}}}{TS_{res}} \right) \tag{3.37}$$

CTS and DTS values can be ambiguous and, therefore, a parameter m is introduced to indicate the number of wrap-arounds. The general equation for both timestamps is [33]:

$$t_{ts}(m) = \frac{timestamp}{TS_{res}} + m \cdot \frac{2^{TS_{len}}}{TS_{res}} \tag{3.38}$$

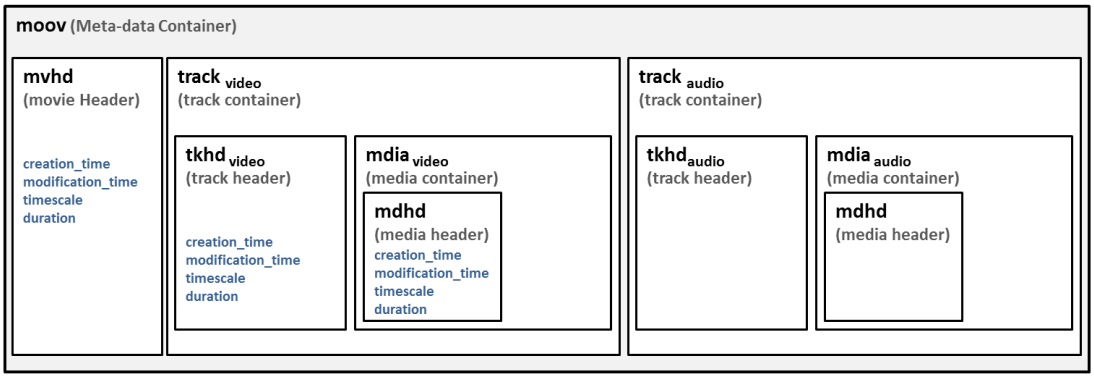


Figure 3.19: ISO File System example with audio and video track with time related fields

Every time a timestamp is received, to prevent these equivocal values, the value m should be the one that minimizes [33]:

$$|t_{OTBestimated} - t_{ts}(m)| \tag{3.39}$$

3.8 ISO Timelines

As seen in Chapter 2 ISO timing is based on boxes which convey information about the media therefore time information and timestamps are coded within boxes. Unlike other MPEG standards there are no clock references related values. In the next sub-section the time related boxes are described.

3.8.1 ISO Time Information

The time information in ISO File formats are found in several boxes such as the Movie Header Box (*mvhd*), Track Header Box (*tkhd*), and Media Header Box (*mdhd*). Table 3.20 contains a summary of the boxes and fields used and Fig. 3.19 shows an ISO file structure with the time fields included for an audio and video stream.

The *mvhd* is the header box of Movie Box *moov*. It conveys the general media-independent information and is related to the entire presentation. Therefore, includes time related information to all media presentation. The structure of Movie Box (moov) and its header are [12]:

```
aligned(8) class MovieBox extends Box('moov'){}

aligned(8) class MovieHeaderBox
           extends FullBox('mvhd', version, 0) {
    if (version==1) {
        unsigned int(64) creation_time;
        unsigned int(64) modification_time;
```

	creation time	modification time	timescale	duration (in timescale units)
Movie Header Box	Movie creation time	Movie modification time	Time units in a second	Movie presentation duration
Track Header Box	Track creation time	Track modification time	Time units in a second	Track presentation duration
Media Header Box	Media creation time (in a track)	Media modification time (in a track)	Time units in a second	Media presentation duration

Table 3.20: Time References within ISO Base Media Format

```
        unsigned int(32) timescale;
        unsigned int(64) duration;
    } else { // version==0
        unsigned int(32) creation_time;
        unsigned int(32) modification_time;
        unsigned int(32) timescale;
        unsigned int(32) duration;
    }
    template int(32) rate = 0x00010000; // typically 1.0
    template int(16) volume = 0x0100;   // typically, full volume
    const bit(16) reserved = 0;
    const unsigned int(32)[2] reserved = 0;
    template int(32)[9]
        matrix ={0x00010000,0,0,0,0x00010000,0,0,0,0x40000000};
    bit(32)[6] pre_defined = 0;
    unsigned int(32) next_track_ID;
}
```

The fields *creation_time* and *modification_time* represent the presentation creation and most recent modification time (units in seconds) since 1st January 1904 in UTC time.

The field *timescale* is the time units, within a second, specified for all presentation whereas, *duration* contains information about the presentation's length in timescale units.

More time fields are found one level below in the hierarchy in the Track Box (*trak*) and its related Edit List Box *elst* and Track Header (*tkhd*).

The edit box (*edts*) is used to introduce presentation offset, this box links the presentation to the media timeline as well as it is an edit list container.

The edit list box (*elst*) provides an explicit timeline link. Every track timeline is defined by an entry, although also could this indicate an empty time. The *track* and the *elst* box structure is the following:

```
aligned(8) class EditBox extends Box('edts') { }
```

```
aligned(8) class EditListBox extends FullBox('elst',version,0) {
    unsigned int(32) entry_count;
    for (i=1; i <= entry_count; i++) {
        if (version==1) {
            unsigned int(64) segment_duration;
            int(64) media_time;
        } else { // version==0
            unsigned int(32) segment_duration;
            int(32) media_time;
        }
        int(16) media_rate_integer;
        int(16) media_rate_fraction = 0;
    }
}
```

The time fields are *media_time* and *segment_duration*. The former indicates the start time of the relative segment, although value (-1) indicates an empty edit. The field *segment_duration* codes, in *mvhd* timescale units, the segment's duration. Finally, the *media_rate* indicates the media play rate.

The last box is the *tkhd* box is defined as:

```
aligned(8) class TrackBox extends Box('trak') { }

aligned(8) class TrackHeaderBox
           extends FullBox('tkhd', version, flags){
    if (version==1) {
        unsigned int(64) creation_time;
        unsigned int(64) modification_time;
        unsigned int(32) track_ID;
        const unsigned int(32) reserved = 0;
        unsigned int(64) duration;
    } else {      // version==0
        unsigned int(32) creation_time;
        unsigned int(32) modification_time;
        unsigned int(32) track_ID;
        const unsigned int(32) reserved = 0;
        unsigned int(32) duration;
    }
    const unsigned int(32)[2] reserved = 0;
    template int(16) layer = 0;
    template int(16) alternate_group = 0;
    template int(16) volume = {if track_is_audio 0x0100 else 0};
    const unsigned int(16) reserved = 0;
    template int(32)[9]
        matrix={0x00010000,0,0,0,0x00010000,0,0,0,0x40000000};
```

```
    unsigned int(32) width;
    unsigned int(32) height;
}
```

The fields *creation_time* and *modification_time* code the track creation and most recent modification time (units in seconds) since 1st January 1904 in UTC time as well as *duration* contains information about the track's length in *mvhd* timescale units.

The structure of Media Box (mdia) and its header are [12]:

```
aligned(8) class MediaBox extends Box('mdia') { }

aligned(8) class MediaHeaderBox extends FullBox('mdhd', version, 0) {
    if (version==1) {
        unsigned int(64) creation_time;
        unsigned int(64) modification_time;
        unsigned int(32) timescale;
        unsigned int(64) duration;
    } else {        // version==0
        unsigned int(32) creation_time;
        unsigned int(32) modification_time;
        unsigned int(32) timescale;
        unsigned int(32) duration;
    }
    bit(1) unsigned unsigned
    unsigned int(5) [3] language;  //ISO-639-2 language code
    unsigned int(16) pre_defined=0;
}
```

The fields *creation_time* and *modification_time* code the media's (within a track) creation and most recent modification time (units in seconds) since 1st January 1904 in UTC time as well as *duration* informs about the media length in *mvhd* timescale units.

3.8.2 Timestamps within ISO

The two boxes related to timestamps are Decoding Time to Sample Box (*stts*) and the Composition Time to Sample Box (*ctts*). The parent for both boxes is the Sample Table Box (*stbl*). The full ISO hierarchy boxes for both tables is found in Fig. 3.20.

The table/box *ctts* aims to index from decoding time to sample number. It is obligatory and only one is required. It contains the decoding time delta and the number or consecutive samples with the same delta. The *entry_count* is the number of entries in the following table, the *sample_count* is the number of samples with the same delta, and finally, the *sample_delta* conveys the samples' delta in the media's timescale. *'By adding the deltas a complete time-to-sample map may be built'* [12].

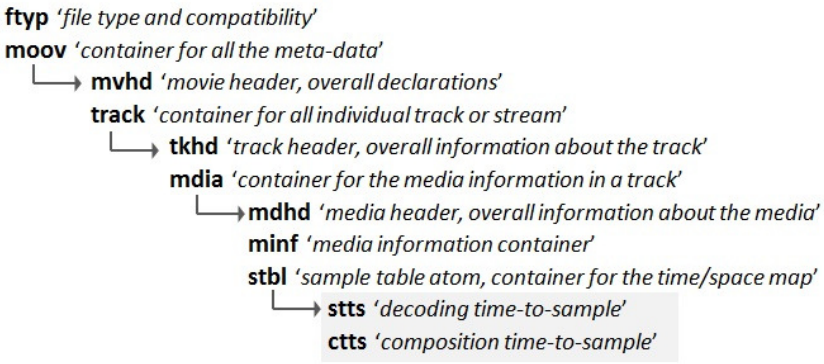


Figure 3.20: ISO File System for timestamps related boxes [12]

The decode time delta's can be derived from this table fields:

$$DT(n+1) = DT(n) + stts(n) \tag{3.40}$$

where n is the sample index and the table entry for the related sample is *stts(n)*, DT(n+1) is the decoding time for the *(n+1)*th and DT(n) is the decoding time for the n^{th}.

The *stts* box structure is:

```
aligned(8) class TimeToSampleBox
           extends FullBox('stts', version = 0, 0) {
    unsigned int(32) entry_count;
    int i;
    for (i=0; i < entry_count; i++) {
        unsigned int(32) sample_count;
        unsigned int(32) sample_delta;
    }
}
```

The *ctts* table/box conveys the difference between the decoding and composition time. It is not mandatory and zero or one boxes can be found in an ISO file. The composition time is always bigger than the decoding time. This box is only required if DTS is not equal to CTS. The *entry_count* codes the number of entries is the following table whereas the *sample_count* signals the number of consecutive samples with the same offset. The offset is:

$$CT(n) = DT(n) + ctts(n) \tag{3.41}$$

where n is the sample index and the table entry for the related sample is *ctts(n)* and CT(n) is the composition time for the n^{th}.

The *ctts* box structure is:

```
aligned(8) class CompositionOffsetBox
```

index	stts		ctts	
	sample_count	sample_delta	sample_count	sample_offset
1	1253	1	13	2
2			1	4
3			1	2
4			1	0
5			1	4
6			1	2
7			1	0
8			1	3
9			1	1
10			1	2

Table 3.21: *stts* and *ctts* values from the track1 (video stream) from ISO example

```
        extends FullBox('ctts', version = 0, 0) {
    unsigned int(32) entry_count;
    int i;
    for (i=0; i < entry_count; i++) {
        unsigned int(32) sample_count;
        unsigned int(32) sample_offset;
    }
}
```

In ISO example in Fig. 2.20 in Chapter 2 there are two media tracks, a video and an audio track (media streams). The video track contains both *stts* and *ctts* boxes whereas the audio contains only *stts* due to the fact that audio decoding and presentation time is always the same. In this particular example *stts* video box one entry mapped to 1253 samples and *ctts* video box has 1059 entries. The audio *stts* box contains 2435 samples. In Table 3.21 the 10^{th} first values of both tables in the examples.

In Table 3.22 the decoding and presentation values are calculated following formulae 3.40 and 3.41.

3.9 MPEG-DASH Timelines

MPEG-DASH, or ISO/IEC 23009-1, is the MPEG Standard for Adaptive HTTP Streaming. It deploys two possible forms of media streaming: on-demand or live Streaming. MPD Static is normally used for On-Demand Streaming where MPD Dynamic is use for Live Streaming. The MPD type dictates the fields' value within the MPD structure.

A high level look at the time fields within an MPD file can be found in Fig. 3.21 and all the fields are described in Appendix C in Table 11. The time fields are distributed within the *MPD*, *period* and *segment* blocks. All the data types follow the XML Schema part 2 Data Types format [88].

	DT(n) DT(n=1)=DT(n)+stts(n)	CT(n) CT(n)=DT(n)+ctts(n)
1	DT(1)=1	CT(1)=1+2=3
2	DT(2)=1+1=2	CT(2)=2+2=4
3	DT(3)=2+1=3	CT(3)=3+2=5
4	DT(4)=3+1=4	CT(4)=4+2=6
5	DT(5)=4+1=5	CT(5)=5+2=7
6	DT(6)=5+1=6	CT(6)=6+2=8
7	DT(7)=6+1=7	CT(7)=7+2=9
8	DT(8)=7+1=8	CT(8)=8+2=10
9	DT(8)=8+1=9	CT(9)=9+2=11
10	DT(10)=9+1=10	CT(10)=10+2=12

Table 3.22: DT(n) and CT(n) values calculated from values in *stts* and *ctts* boxes from the track1 (video stream) from ISO example

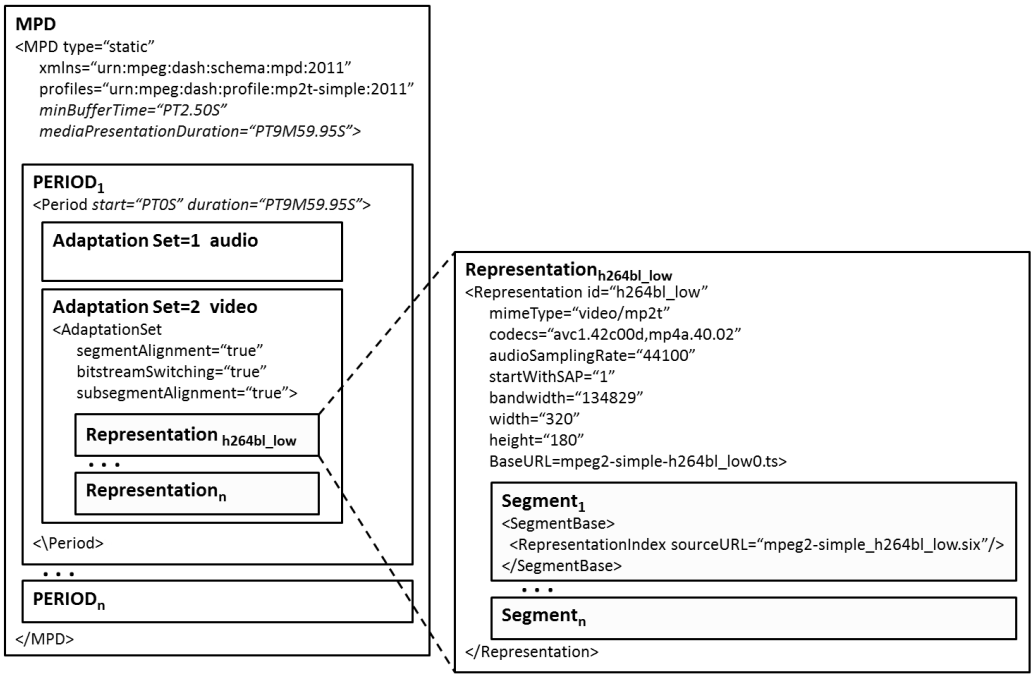


Figure 3.21: MPD example with time fields from [89]

The time fields within the *MPD* element establish the general requirements for the media delivery linked to the MPD file delivered to the client.

Within *period* only two fields are found, *start* and *duration*. Both outline timing information for a defined period. The former indicates the start of the period and the latter its duration. If *start* element is not defined, it can be calculated form the *start* and *duration* from the previous period. Moreover if *start* element is missing from the first period, this indicates the MPD is

```
<Representation id = "1" mimeType = "video/mp2t"
    width = "320"
    height = "180"
    BaseURL = mpeg2-simple-h264_low.ts>
    <SegmentBase timescale = "1000"
        duration = "10000"
        startNumber = "0">
        <RepresentationIndex ="video320x180_48kbps_Index.didx">
        </RepresentationIndex>
    </SegmentBase>
</Representation>
```

Figure 3.22: MPD example with time fields using Segment Base Structure from [89]

```
<Representation id = "1" mimeType = "video/mp2t"
    width = "320"
    height = "180"
    BaseURL = mpeg2-simple-h264_low.ts>
    <SegmentTemplate
        timescale = "1000"
        duration = "10000"
        startNumber = "0"
        media = "video320x180_48kbps$Number$.ts"/>
        <RepresentationIndex="video320x180_48kbps_Index.didx">
        </RepresentationIndex>
    </SegmentTemplate>
</Representation>
```

Figure 3.23: MPD example with time fields using Segment Template from [89]

type Static and the initial value of beginning of first the period is zero [59].

Within every *segment* there are three time fields, *timescale*, representing the time scale in units per second, *duration*, indicating the segment time duration, and *presentationTimeOffset*, shows the presentation offset from the beginning of the period's start (default value is zero) [59].

There is an extra system to include timelines within the segments, via the *segmentTimeline*. This timeline includes fields such as t, d, and r. Values t, d relate to the time and duration, respectively. Finally, r indicates the number of segments which apply the d value.

There are three MPD examples shown in Fig. 3.22, Fig. 3.23 and Fig. 3.24. In Fig. 3.22 there is an example of *segmentBase*. In Fig. 3.23 an example of *segmentTemplate* is shown. In both cases the fields *timescale* and *duration* are included. Finally, in Fig. 3.24 an example of the *segmentTimeline* is found with all its fields.

There are multiple examples of the implementation of multimedia delivery via MPEG-DASH over Internet providing tools for media synchronisation. E.g., MPEG-DASH is used to design a Web-based Synchronization Framework (WMSF) to test two scenarios Video Wall ('*a tiled video where an independent screen represents each tile*' [90]) and Silent TV ('*a TV screen and multiple second screen devices, e.g., phone or tablet*' [90]).

```
<Representation id = "1" mimeType = "video/mp2t"
    width = "320"
    height = "180"
    BaseURL = mpeg2-simple-h264_low.ts>
    <SegmentTemplate
      timescale = "1000"
      startNumber = "0"
      media = "video320x180_48kbps$Number$.ts"/>
      <SegmentTimeline>
          <S t = "0" d = " " r = " " />
      </SegmentTimeline>
      <RepresentationIndex="video320x180_48kbps_Index.didx>
      </RepresentaionIndex>
    </SegmentTemplate>
</Representation>
```

Figure 3.24: MPD examples with time fields using Segment Timeline from [89]

3.10 MMT Timelines

As seen in Chapter 2.3.6 MMT is divided into different layers. The timing structure proposed is based on this structure, D-Layer, E-Layer and the S-Layer. This timing model proposes time fields within the E-Layer and D-Layer [91].

The timing model aims to provide a common timing information from sender to receiver in the encapsulation and delivery process. The E-Layer should provide media sync and timing information to facilitate media play-back at user-side, whereas the D-Layer should provide delivery timing information and capabilities to re-adjust timing associations to cope network jitter [91].

Within the E-Layer, fields include *SamplingTime*, *DecodingTime*, *RenderingTimeOffset* and *NTPtime*. These fields provide the tools to enable media sync at receiver-side.

SenderProcessingDelay, *DeliveryTime*, *ArrivalTime*, and the *TransmissionTime* are all proposed within the D-Layer for the media delivery. In Fig. 3.25 the timeline from sender to receiver and the time values can be seen. Fig. 3.26 includes the MMT timeline model between a MMT sender and receiver. Finally, in Fig. 2.24 in Chapter 2 outlines the MMT architecture at a high level with all time fields located in the related layer.

Two options are proposed to provide timing within MMT. One is to UTC sync every element within the delivery path via an NTP server. The main advantage is that all elements would have access to the clock references. The other provides for the addition of in-line clock references to make MMT more widely deployed [45].

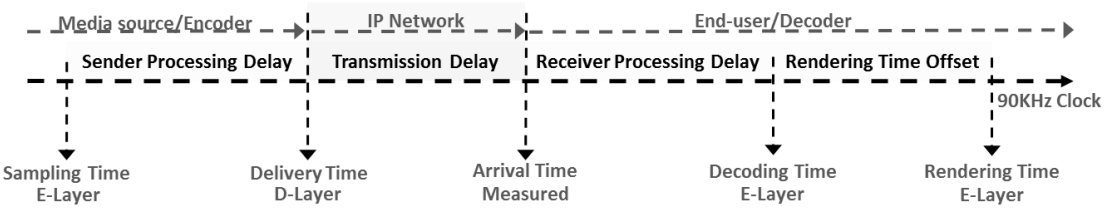


Figure 3.25: MMT Timing system proposed in [91]

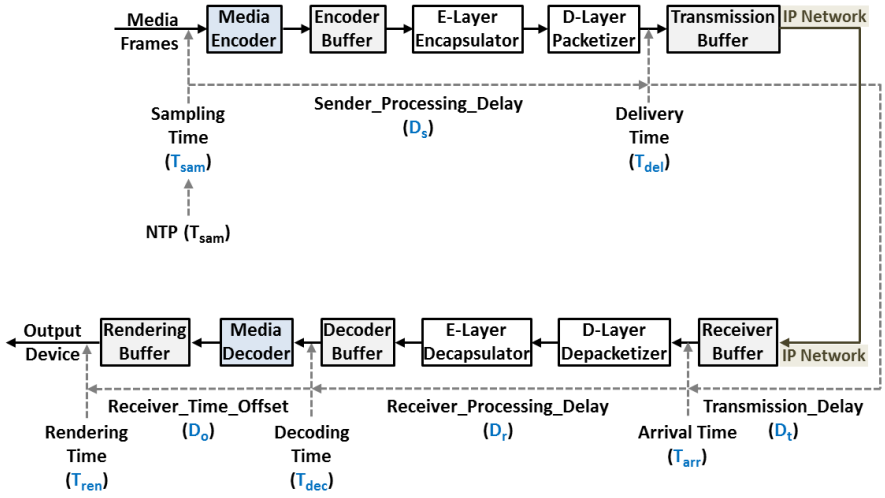


Figure 3.26: MMT model diagram at MMT sender and receiver side [91]

3.11 Multimedia Sync. Solutions and applications

3.11.1 Media Delivery

Begen describes media streaming techniques in depth. He differentiates between the two main media delivery methods, namely push and pull media streaming. Push streaming relates to RTP/UDP streaming whereas Pull streaming is Adaptive HTTP streaming via TCP. One of the key differences between the two techniques is that push-based streaming supports IP multicast delivery whereas pull-based streaming is only delivered via IP unicast [92].

Push based streaming basically uses RTP as a media delivery and RTSP as a session control protocol. The session state is retained by the server which updates with any session-state variations from the client.

Push based streaming accomplishes smooth play-out and play-back due to its capability to adjust transmission rate and by monitoring client's bandwidth and buffer levels. It streams at the appropriate media encoding bitrate to match the client's media consumption rate.

The media server thus accommodates the bitrate stream to the network and receiver conditions. For example, it may shift to a lower-bitrate stream to prevent buffer overflow and change

to a higher-bitrate when buffer conditions allow. The client provides bandwidth monitoring and network metrics to the server such as network jitter, Round-Trip Time (RTT), and packet loss to server.

Pull-based streaming is HTTP based and thus, does not have issues traversing firewalls and NAT services and the state information is the minimum required. This makes the solution more scalable.

The client plays an important role by being in charge of requesting the media from the server. Sever provides bitrate adaptation to prevent buffer overflow or underflow when it is requested by the client.

There are more concepts included in the media delivery. For example, it differentiates between streaming to a home client from a home server, streaming to a home client from an Internet server, streaming to a home client from a managed server and streaming to a home client via P2P delivery [93].

Home client from a home server is not very common due to the technical knowledge needed. Streaming to a home client from an Internet server only uses pull-based streaming whereas streaming to a home client from a managed server is able to use both pull and push-based streaming [93].

A deep study of Internet Video Streaming discerns between three stages. Firstly, client-server video streaming, using RTP, secondly, P2P video streaming using P2P protocols, and finally, HTTP video streaming in the cloud [94].

Client-server video streaming research is mainly focused on RTP. The main areas of research area are rate control, rate sharing, error control and proxy catching. Finally, RTP facilitates IP multicasting which is mainly used in IPTV media platforms [94]. As seen in Chapter 2.4.1.

P2P video streaming is based on the concept that hosts, called peers, have dual functions: they work as clients and servers in unison. The two main advantages are the lack of a network infrastructure and peers functionality of simultaneously downloading and uploading. However, the main inconvenience is the need for special software to run the P2P protocols [94].

The last technique is HTTP video streaming in the cloud (also called HTTP Adaptive Streaming). The main principal of this technique, seen in Chapter 2, involves downloading of small chunks of media data via HTTP. It is the principal video streaming system used nowadays over the Internet [94].

Service Levels Agreements (SLA) are the specified requirements the consumer of services expect from the service providers.

Due to user expectations, SLAs have more restricted requirements in IPTV than Internet TV. The three key direct areas related to SLA metrics are Network Delay, Network Jitter and Packet Loss [95].

Network Delay measures the residency time of an IP packet in the IP network. It is also called one-way network delay. The elements impacting on the Network Delay are:

- propagation delay through the network path

- switching and queuing delays at network elements on the path
- serialization delay

The principal impact of the network delay for TV/video is the *channel-change-time*, also named *finger-to-eye*. Service providers aim for a maximum of 100ms to achieve an overall 2s channel-change-time.

Network Jitter is the difference in network delay for two successive packets. De-jittering buffers are used to eliminate the network jitter. In such a scenario the buffer size affects the performance, a smaller buffer size can result in buffer underflow whereas a bigger buffer size can add unnecessary end-to-end (e2e) delay.

Packet Loss is the number or percentage of packets that don't arrive at the expected time at the receiver. The factors impacting on Packet Loss are: Congestion, Lower-layer errors, and Network element failures. Packet loss can also occur at the end receiver where packets either overflow or arrive too late.

Network Delay, Network Jitter and Packet Loss can have an impact on the video quality resulting on artifacts such as slice error, blocking or pixelization, ghosting and freeze frame [96].

Slice error occurs when an IP packet is dropped at the network. The result is a small error in the picture. It could be propagated within the GOP but it gets fixed when an unimpaired I-frame arrives.

Blocking or pixelization occurs when an I or P-frame is dropped in the network. Therefore, all further frames will miss important information for decoding. The impact is bigger than a slice error as the slice error gets fixed when an unimpaired I-frame is received.

Ghosting occurs when an I-frame or a large number of slices close to a scene change are lost. Like the slice error and pixelization, this gets fixed when an unimpaired I-frame is received.

Finally, frame freeze occurs when multiple frames are lost. The last frame is displayed until new frames are received.

3.11.2 Applications

Multimedia sync is a broad term that describes a range of scenarios. One such application of particular interest to this thesis is the sync of multiple media formats delivered from multiple sources to a unique user.

One practical application is the solution presented in that addresses the problem of delays on live program subtitles at user-side. Needless to say, there is no problem in subtitling of pre-recorded programs as the subtitle stream is multiplexed within the MP2T stream with the correct timestamps [97]. The case study tackled the issue of live subtitles programs where the audio is not predictable such as live programs [97]. Usually the process of subtitling these programs involves a series of steps including speech to text that generates the subtitles from the audio and a person who then proof reads the text to fix possible errors. As a last step the subtitle is inserted in the MP2T stream. As such, this process can result in subtitles that are

out of sync at play-out.

The solution proposed a fix that involves the delivery of a broadcast TV via IPTV with the necessary delay added to compensate for the subtitle delay generation. The timestamping of the subtitles is inserted in the multiplexed MP2T at the time when they should be displayed [97].

Note that the users not requiring the subtitles are able to receive the TV program via broadcast (DVB) but the users who require the subtitles can watch the program via an IPTV channel with a few seconds delay but with the live subtitles synchronised with the Live TV program.

Another application of media sync is proposed [98] [99]. The solution takes advantage of the HbbTV ability to use a single receiver at user-side for broadcast and broadband TV. The solution aims to free up the broadcast resources by streaming via broadband those channels with a reduced audience. Media sync is used to switch the same TV program emission from broadcast to broadcast delivery.

In this scenario, the broadband full dual channel communication provides feedback to the media server about the number of users watching a specific TV program. When the number of users is above a predefined level the system sends the TV stream via broadcast but when user number decrease then the TV stream is delivered via broadband. This requires seamless switching between broadcast and broadband delivery which needs to be performed using media sync so the user's play-out does not get affected and users are not aware of any change in the delivery platform.

This system enables TV systems to adapt their delivery technology based on audience feedback. It is further developed by providing time shifting control delivery. The system pre-stores TV programs based on user's preferences thus the play-out time differs from delivery time [99].

Ciril Concolato also proposes a very good example of media sync applications and solutions where he studies the MPEG-DASH media delivery with Rich Media Data (audio, video, graphics, textual meta-data, animations, etc). He describes how within a MPEG-DASH session, the Rich Media Services are coded to guarantee tight sync with the MPEG-DASH audio and video data [100].

Concolato also presents synchronised delivery over broadband and broadcast networks. He studies the identification of content related media from different networks, the synchronisation and re-sync to adapt to network conditions. The study is performed in Hybrid Delivery systems and presents the idea of synchronising a broadcast FM station with a broadband delivered MP2T stream [101].

Concolato, after explaining the inconvenience of different bootstrapping techniques, uses audio channel bootstrapping information conveyed within Radio Data System (RDS) called Open Data Applications (ODA).

In the first place, the radio set-up is performed, then, in second place the MP2T stream is fetched. Only then does the synchronisation take place. The timelines used are the TDT from the broadband MP2T stream and the UTC Clock Time (CT) from the broadcast Radio

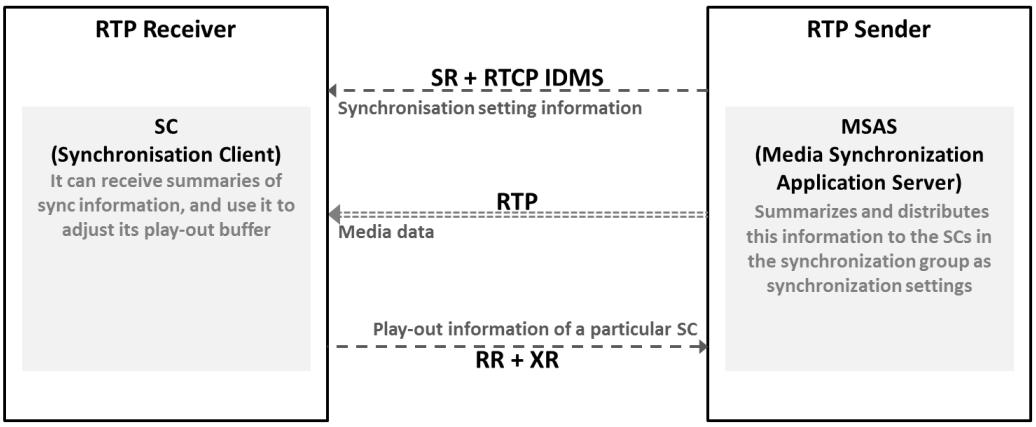


Figure 3.27: IDMS Architecture Diagram from [102]

channel.

Concolato explores the sync between two broadband MP2T streams sync, which is done via the TDT and PCR values of both video streams.

3.11.3 Inter-destination Media Sync via RTP Control Protocol

RFC 7272 standardises a tool to provide IDMS by the use of RTCP for IDMS by means of the definition of a new RTCP packet type and a new RTCP Extended Report Block type. IDMS is *'the process of synchronising play-out across multiple geographically distributed media receivers'* [102]. As an example, IDMS is adapted to MPEG-DASH to provide inter-synchronisation playback among geographically distributed peers [102].

IDMS applications examples are quite varied in scope and include Social TV, Video Walls and Network loudspeakers. Social TV is the scenario where multiple users, from different locations, are sharing the play-out of a unique media stream and due to synchronised play-out, they are able to comment on the play-out via a text platform. Video walls is the display of multiple TV screens together to become a unique large screen. Finally, the scenario of multiple network loudspeakers used in large rooms or large venues such as stadiums presents yet another scenario [102].

The IDMS architecture has two main components, the Media Synchronisation Application Server (MSAS) and the Synchronisation Client (SC). The latter reports back to the MSAS on the arrival and play-out times via the RTCP IDMS XR reports. The MSAS collects this information from all the SCs. Once all information is collected and summarized it is sent back to all SCs via the RTCP IDMS Settings message [102].

The key features are the RTCP XR Block packet for IDMS, to send the SC play-out information, and the RTCP Packet Type IDMS Settings, to send synchronising settings information. In Fig. 3.27 an example of IDMS architecture is shown. It shows the SCs sending the RTCP

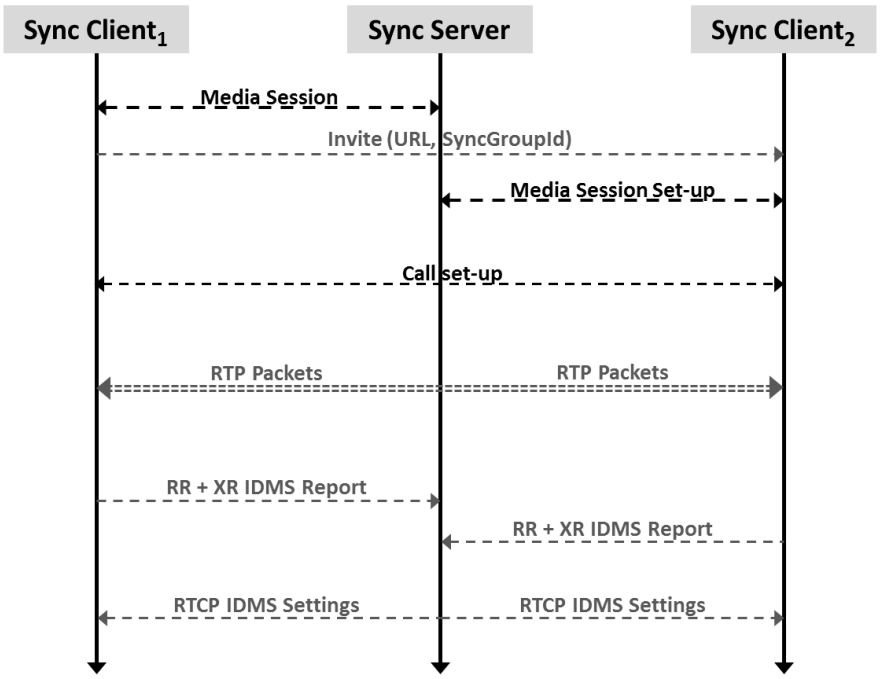


Figure 3.28: Example of a IDMS session. Figure 1 in [102]

RR and XR report packets to the MSAS and the MSAS sending each of the SCs the RTCP SR and IDMS Settings packets [102].

In Fig. 3.28 an example of IDMS media session is presented. Once the media session has been set-up and RTP media packets are being delivered to clients, the RTCP RR and XR packets are sent to the MSAS and the MSAS responds sending the RTCP IDMS Settings packet to the SCs [102].

The information within the RTCP XR Block packet is conveyed in the *Packet Received NTP timestamp*, *Packet Received RTP timestamp*, and *Packet Presented NTP timestamp* as seen in Fig. 3.29.

The information within the RTCP IDMS Settings packet is conveyed in the *Packet Received NTP timestamp*, *Packet Received RTP timestamp*, *Packet Presented NTP timestamp* as seen in the RTCP XR block structure in Fig. 3.30.

The SC reports back to the MSAS on the received and presented NTP timestamps together related to the RTP timestamps. The IDMS sync aims to sync the packet arrival, decoding and rendering times, with all SCs having the same buffer settings. The RTCP IDMS attribute in SDP is used to indicate the use of this solution and to transmit synchronisation group identifiers used by the clients to join [102].

Adaptive Media Play-out (AMP) has been proposed to achieve better results for IDMS. AMP can ensure that play-out discontinuities are minimised in IDMS when buffering techniques are not sufficient in congested environments [103]. Moreover, the benefits of AMP based

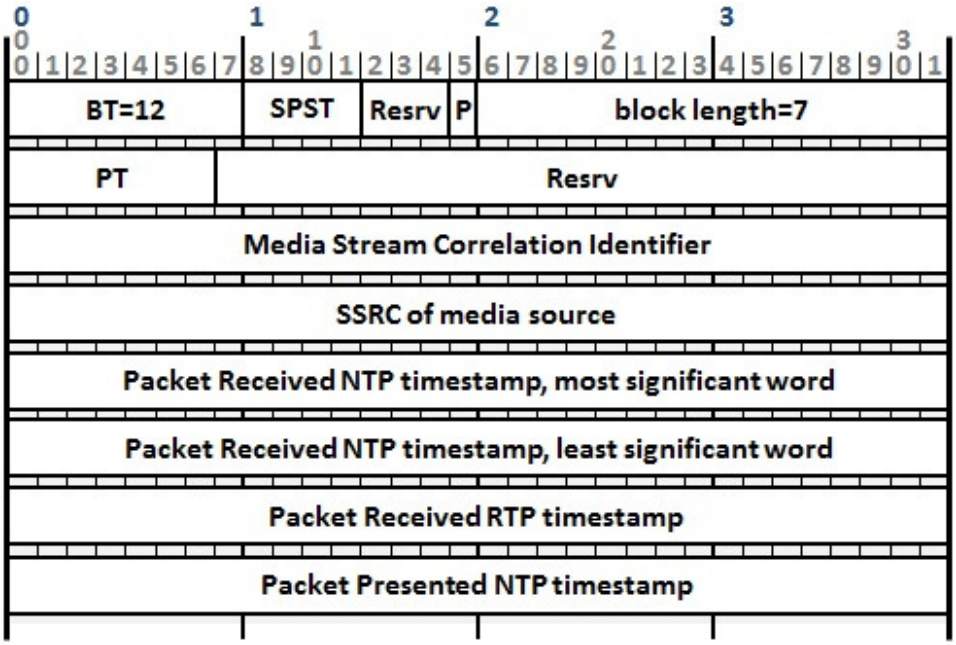


Figure 3.29: RTCP XR Block for IDMS [102]

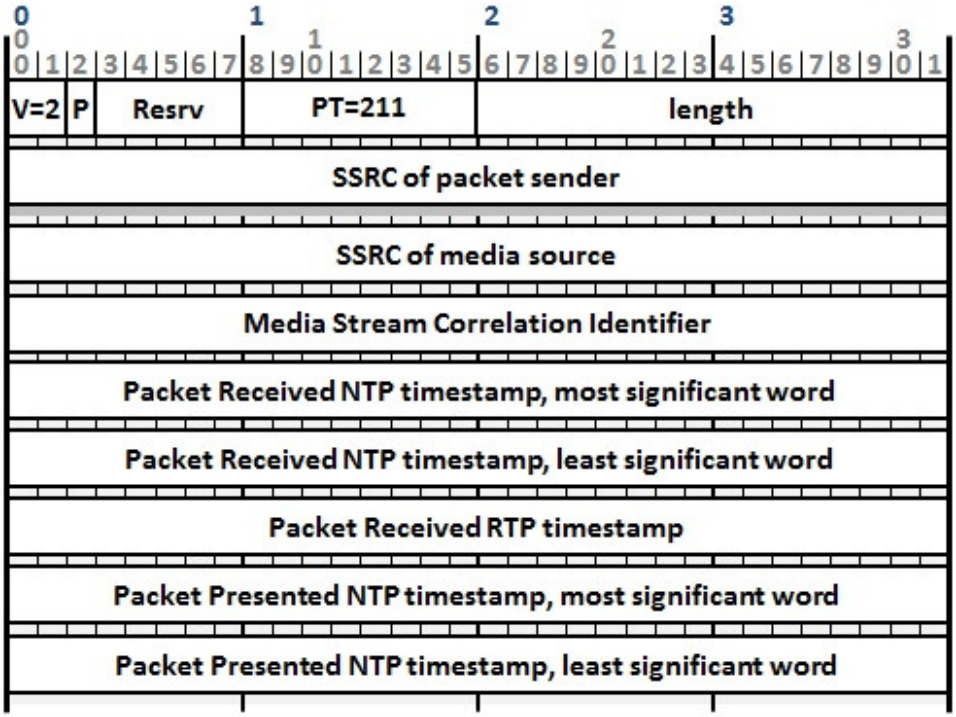


Figure 3.30: RTCP Packet Type for IDMS (IDMS Settings) [102]

on the modification of the playback rate in IDMS have been studied and metrics of the impact of the variation of playback rate have been established [104].

Context-aware adaptive media play-out can be used to adjust the play-out rate to control the synchronisation, in other words, the play-out rate can be adjusted to control the synchronisation [105]. The sync method implies that the play-out rate can be modified in such a way that it is not noticeable by the user. It is based on the hypothesis that '*high motion scenes with a low volume in audio can be slowed down and scenes with low motion and low volume are candidates for increasing the play-out rate*'[105]. An algorithm is presented to analyse the lower and upper restrictions of video (motion vectors between consecutive frames) and audio (Root Mean Square of audio frames over time). MPEG-DASH is also proposed for further assessment of the algorithm implementation within a media player prototype [105].

3.11.4 Multimedia Sync. HBB-NEXT Solution (Hybrid Sync)

HBB-NEXT is a now completed EU funded project (EC FP7 Project 2010-2014) which intended to enrich features in HbbTV such as *'multiple device support, social media integration, personalised user/group experience'*.

The solution proposed by HBB-NEXT for Multimedia Sync represents the application of ETSI 102 823 [106] and was presented recently to HbbTV. This standard specifies the carriage of synchronised auxiliary data within DVB MP2T streams. Details of this project can be found in the HBB-NEXT evaluation technical report [107] [108] and the HBB-NEXT Report on User Validation reports [109] and prototypes using the specification have been developed and proven [110] [111]. The test-bed syncs a DVB Transport Stream with a sign language stream, both video streams, with both displayed on a single screen [110]. The test-bed is extended to sync the sign language video and audio with IP subtitles and also examines inter-destination sync IDMS (as well as inter-media sync) [111]. To achieve inter-media sync, the system extracts PTS timestamps from a DVB broadcast stream, maps this to wall-clock time using the ETSI standard and this information is carried within the DVB stream by replacing some of the stuffing bytes. This is termed the master stream. The slave stream, in this case a signed video stream, is carried using MPEG-DASH and the MPD file is used to indicate the mapping between segments and wall-clock time. In Fig. 3.31 the process of timestamping the PTS coded in the MP2T packet conveying the descriptors is shown. In Fig. 3.32, a sample of MPEG-2 PSI and DVB-SI tables using the solution is shown, in particular, the mapping is implemented using the broadcast timeline descriptor field.

Prototypes in both test-beds use MPEG-DASH for the video delivery of DVB MP2T streams which play the master role while others streams adapt the play-out to sync to the master stream. The MPEG-DASH media server is the slave whereas the DVB MP2T media server acts as the master server.

The timing information or descriptors are packetised into MP2T packets. In Table 3.23 all descriptors used are listed, including the minimum repetition rate of the descriptors. In order

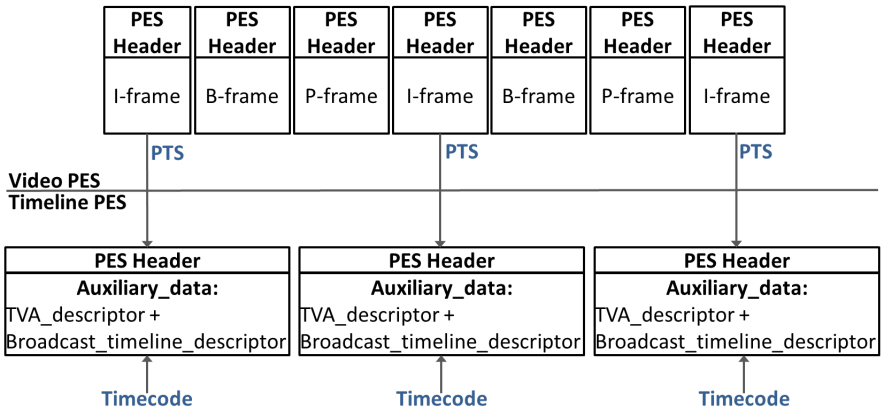


Figure 3.31: High Level broadcast timeline descriptor insertion [110] [111]

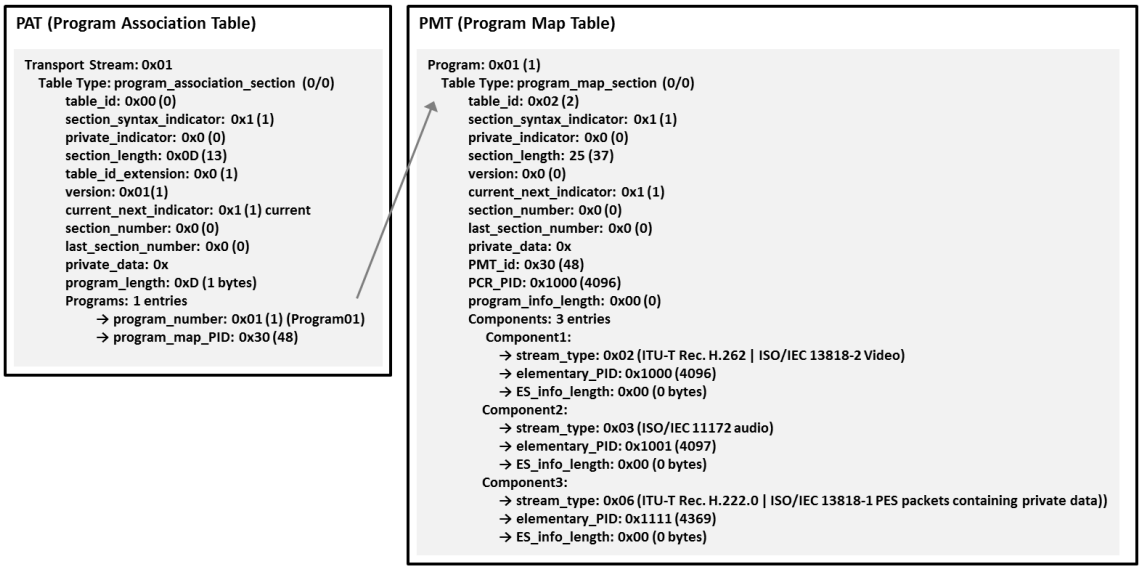


Figure 3.32: High Level DVB structure of the HbbTV Sync solution

to convey this auxiliary data, the following values in the following fields are manipulated [106]:

- *Stream_type*: 0x06 within the MP2T header indicating *ITU-T Rec. H.222.0 — ISO/IEC 13818-1 PES packets containing private data*
- *Stream_id*: 1011 1101 (0xBD) within PES header indicating stream coding *private_stream_1*
- *Data_alignment_indicator*: 1 within PES header
- *PES_packet_data_byte*: Auxiliary data structure bytes/information
- *PTS*: PTS is encoded within the MP2T packet

Tag Value	Identifier	Minimum Repetition Rate	Structure found in Table
0x00	DVB reserved	-	
0x01	TVA_id_descriptor	2s	23
0x02	broadcast_timeline_descriptor		24
	type=0 (direct encoding)	2s	
	type=1 (offset encoding)	5s	
0x03	time_base_mapping_descriptor	5s	25
0x04	content_labelling_descriptor	5s	26
0x05	synchronised_event_descriptor	-	28
0x06	synchronised_event_cancel_descriptor	-	29
0x07-0x7F	DVB reserved	-	
0x80-0xFF	User private	-	

Table 3.23: Descriptors for use in auxiliary Data Structure. Table 3 in [106] includes the minimum repetition rate of the descriptors

There are two situations when *stream_type* and *stream_id* may not be enough to identify a specific stream. First when there is more than one DVB service conveying synchronised auxiliary data, and second when it could be used for other applications. One possible way to differentiate is via the *component_tag* field within the PMT Table [106].

The synchronised auxiliary data within DVB is indicated within the *ES_info* in the PMT Table (See above Fig. 3.32). The relevant fields are:

- *metadata_application_format*: The same value as the content labelling descriptor instance
- *content_reference_id_record_flag*: 0
- *content_time_time_base_indicator*: 0

More details on the Auxiliary Data Structure is depicted in Table 22 (Appendix F).

3.11.4.1 TVA_id Descriptor

This descriptor provides the means to relate metadata to the timeline via the *TVA_id*. The structure of the Broadcast Timeline Descriptor can be found in Table 23 (Appendix F).

3.11.4.2 Broadcast Timeline Descriptor

This descriptor provides a link between a specific point in the broadcast with a wall-clock time value. There are two types of broadcast timelines, direct broadcast timeline, *broadcast_time_type*=0, and the offset broadcast timeline *broadcast_time_type*=1.

In the direct broadcast timeline the broadcast timeline descriptor encodes the absolute time values. The offset broadcast timeline descriptor encodes an offset time value applied to a direct broadcast timeline. The structure of the Broadcast Timeline Descriptor can be found in Table

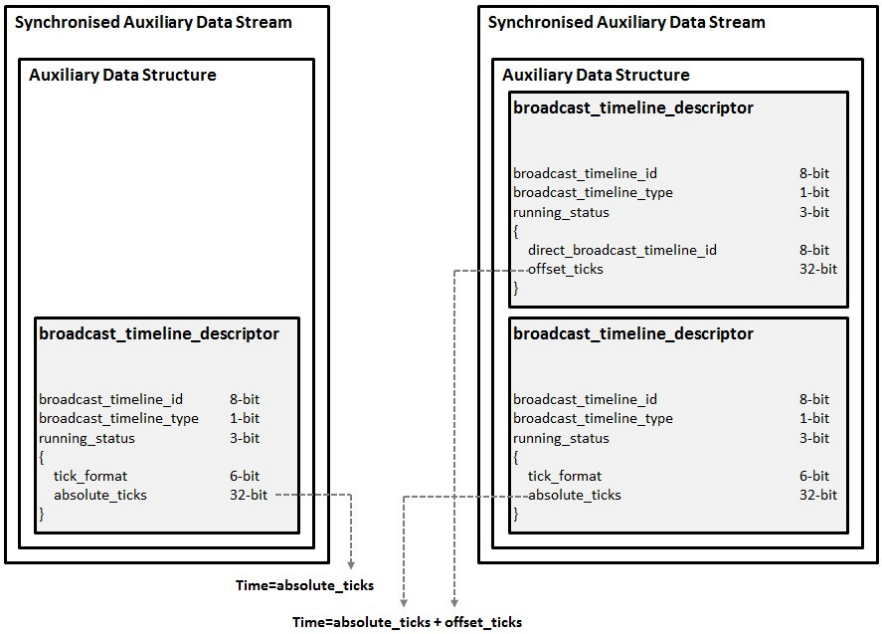


Figure 3.33: Links between timeline descriptors fields to implement the direct, from Fig. D.1 in [106], and offset, from Fig. D.2 in [106], broadcast timeline descriptors

24 (Appendix F). Fig. 3.33 shows the links between two broadcast timeline descriptors to implement the offset type.

With HBB-NEXT prototypes, the tick rate was set at 1000Hz and a start value of zero was given to the start of the master video. Similarly the first segment of the slave MPEG-DASH signed video was given a start time of zero, thus facilitating sync. However, it is important to note that these were not traced back to UTC, and thus, whilst the system outlines the huge potential of inter-media sync, it does not explicitly address this challenge of mapping both streams to UTC.

3.11.4.3 Time Base Mapping Descriptor

This descriptor is used to link a broadcast timeline descriptor with an external time base. The structure of the Broadcast Timeline Descriptor can be found in Table 25 (Appendix F).

3.11.4.4 Content Labelling Descriptor

This descriptor is used to label/identify a content item. Moreover, it provides the means to link the item of content with a broadcast timeline via the identifier. It can be coded within the same or different auxiliary data structure. The structure of the Broadcast Timeline Descriptor can be found in Table 26 (Appendix F) and the private data structure in Table 27 (Appendix F). In Fig. 3.34 shows the first case, same auxiliary stream, and Fig. 3.35 shows the content

Figure 3.34: Example *content_labelling_descriptor* using *broadcast_timeline_descriptor*. Fig. D.3 in [106]

labelling descriptor in a different auxiliary stream than the broadcast timeline descriptor.

3.11.4.5 Synchronised Event Descriptor

This is the tool which facilitates sync of an application-specific event with another broadcast stream component, in this case, a synchronised event. The synchronised Event Descriptor needs to be conveyed within the same Synchronised Auxiliary Stream. The structure of the Broadcast Timeline Descriptor can be found in Table 28 (Appendix F).

3.11.4.6 Synchronised Event Cancel Descriptor

It is the tool to cancel the sync of an Event which is pending, in other words, synchronisation will be performed in the future. The structure of the Broadcast Timeline Descriptor can be found in Table 29 (Appendix F).

3.12 Summary

This chapter presented a range of topics relating to the core research area of multimedia synchronisation. It firstly looked at the relationship between synchronisation and timing and its basis in clocks. Achieving and maintaining clock synchronisation is key to media synchronisation but is a non trivial task. The chapter then detailed the differing media sync types, sync

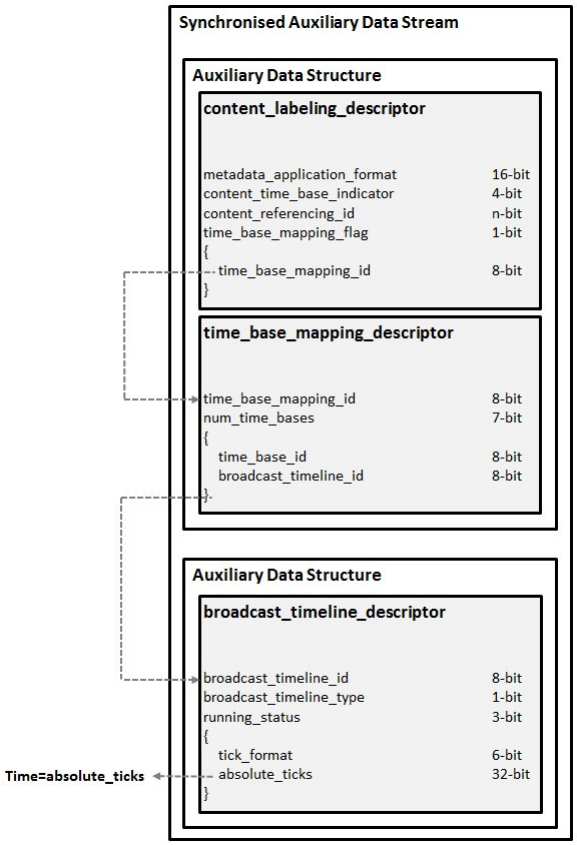


Figure 3.35: *Content_labelling_descriptor* using *time_base_mapping* and *broadcast_timeline_descriptor* example. Fig. D.4 in [106]

thresholds, and time distribution protocols such as NTP, GPS and PTP.

Despite the variety of media containers used and described, a common requirement to perform media synchronisation relates to clock references and timestamps in order to map timelines. In this chapter, a deep analysis of timeline implementation was undertaken to facilitate media sync at client-side. Although the most common media container is MPEG-2 Transport Streams (used in broadcast and broadband technologies), other newer formats are also described such as MPEG-4, ISO BMFF and the latest MMT. MPEG-DASH was also studied, although it could be classified more as a transport media protocol than a media container, with Adaptive HTTP Streaming being the most used media streaming delivery method over the Internet. Finally, a review of some of the more relevant media sync solutions was undertaken. Special attention has been paid to Inter Destination Multimedia Synchronisation (RFC 7273), the solution proposed in ETSI 102 034 and the solution proposed by HBB-next (Hybrid Synchronisation).

Despite the recent developments in media synchronisation summarised in this chapter, a significant gap in the State of the Art (SOTA) exists relating to finely synchronised multi source content delivered to a single device. Solutions such as IDMS whilst very useful are based on

synchronising similar content on multiple devices, whereas HBB-NEXT, whilst closer to the research proposed in this thesis, does not address finely grained synchronisation requirements and the integration of multiple streams into a single stream. This gap informs the remainder of the thesis ultimately resulting in the prototype design as detailed in the next chapter.

Chapter 4

Prototype Design

In the previous chapters, the background material relating to media sync and timelines within different MPEG standards was presented along with the State of the Art (SOTA) in media synchronisation. Whilst much interesting work has been done, the issue of fine grained multi source synchronisation raises many challenges and has not yet been tackled. This chapter focuses on the key thesis contribution. It firstly reinforces the key research Questions, and presents a very high level architecture of a generic solution. It then focuses in on the particular case study and details the methodology and the proof-of-concept design to implement and test the solutions. The discussion on prototype design includes the technology and media files used, the media delivery protocols, the prototype's high level description and the scenarios tested. It also describes the techniques used to accomplish the following: the bootstrapping, sport's events initial sync, MP3 clock skew detection and correction, MP2T clock skew detection and, finally, the multiplexing of video and audio streams into a single MP2T Stream.

4.1 Research Questions

It is useful at this stage to revisit the key research questions. As discussed, they relate to media sources, encoding standards, and delivery platforms, and are expressed as follows:

- Given the variety of current and evolving media standards, and the extent to which timestamps are impacted by clock inaccuracies, how can media synchronisation and mapping of timestamps be achieved?
- Presuming that a mapping between media can be achieved, what impact will different transport protocols and delivery platforms have on the final synchronisation requirement?
- What are the principal technical feasibility challenges to implementing a system that can deliver multi-source, multi-platform synchronisation on a single device?

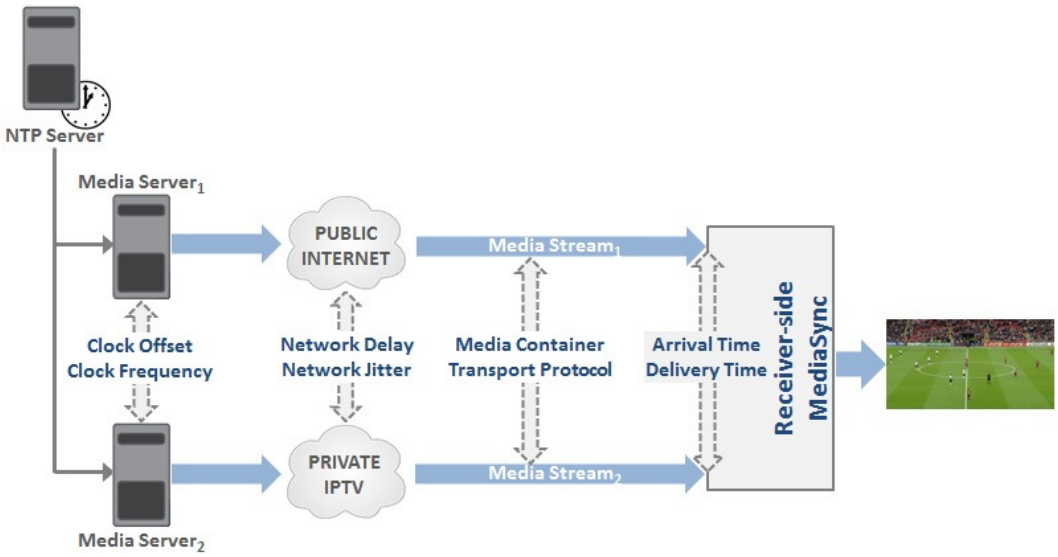


Figure 4.1: High Level Diagram of System Architecture

4.2 High Level Solution Architecture

This section presents a generic solution architecture at a high level. It is depicted in Fig. 4.1. Its principal components are:

- Multiple media sources, each using perhaps different encoding details.
- Transport of the media using a variety of transport protocols and delivery platforms.
- Delivery to a single consumer device whereby the media streams are decoded, buffered as required, time aligned (with skew detection/compensation), and integrated into a single stream for play-out.
- A common time standard across the complete architecture.

Regarding the latter point, having a system-wide time standard facilitates media timestamping at source and media timestamping within transport protocols if required which thus facilitates time alignment at destination, as well as skew detection and compensation. Having time synchronisation available at receiver also facilitates delay calculations which can be important for delay sensitive applications. As outlined earlier, the multiple media source clocks will be affected to varying degrees by clock offset and/or clock skew issues.

4.2.1 From High Level to Prototype

The prototype solves the main functional issues related to the idea of synchronising content through use of NTP. It is a widely used global time distribution protocol and is used by the

transport protocol RTP/RTCP to map between system and media clock timestamps as detailed later. It is also used to determine when on client side to start the synchronisation and integration process for the two media streams, video and audio. There is currently no standard technical tool to ensure that media servers are using NTP correctly for synchronisation but the prototype assumes this. Furthermore, the client side also uses NTP to implement the MP3 audio clock skew detection and correction when required, as well as the MP2T clock skew detection.

Regarding the IP delivery platform, having different platforms can result in very different network delay and network jitter. Using different media containers and transport protocols means that the different media may have different arrival/delivery time at the receiver-side affecting the media synchronisation process.

For the prototype, the TV is delivered via DVB-IPTV platform and Internet Radio via Internet. The prototype synchronises the media from these different IP Networks by using the RTP Transport Protocol which provides the tools via RTCP to synchronise the media streams at client-side by providing NTP values related to RTP timestamps.

Finally, the media containers used in the prototype involve the use of MP2T stream with MPEG-2 PSI and DVB-SI tables for video, and MP3 for Internet Radio. Synchronisation and clock skew issues are resolved between the two streams by detecting skew rate of both streams relative to UTC (via NTP) and then correcting the MP3 stream such that it matches the MP2T skew. The last step in the prototype involves the integration of the skew free audio into the MP2T stream for a single play-out in the media player.

4.3 Detailed Prototype Description

The prototype requires two media streams, one video stream (with embedded audio) via IPTV and one radio stream via Internet TV. The video is stored on a server in MP2T format and streamed to the client via RTP/UDP simulating an IPTV environment. The Radio audio is stored on a server in MP3 format and streamed to the client via RTP/UDP. Both streams are processed on the client and integrated/synchronised into a single MP2T stream. In the prototype, the final stream is simply stored locally and played back for validation using VLC. In a real environment, the integrated stream would, of course, be played out in pseudo-real-time.

There are two possibilities when multiplexing a TV channel with a Radio channel:

- Audio channel substitution → Easier implementation but user has no longer access to the original audio
- Audio channel addition → Multiple audio user selection between original and added audio, with the additional overhead of a more complex implementation

As initial work for the prototype, an MP2T/DVB and MP3 media analyser has been deployed. The server streams the media streams and the related client analyses at socket layer the packets

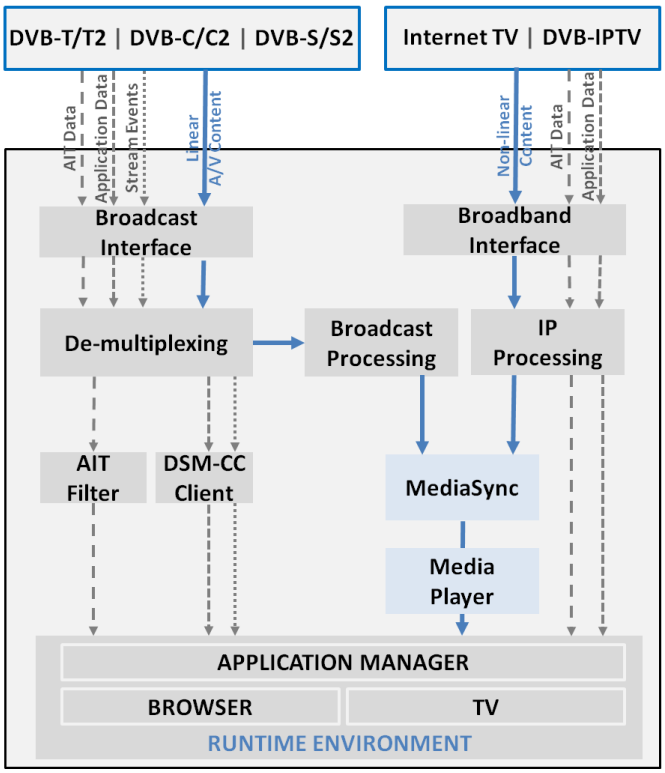


Figure 4.2: Prototype illustrated within HbbTV Functional Components. Figure 2 in [22] with added proposed MediaSync module

received. A reliable client-side analyser was needed because the free-ware media analysers found in the Internet only work on MP2T stored files.

In the prototype there are four threads, two for streaming the media files, at the server-side and two for reading/processing the media files at client/receiver. The MediaSync module shown in Fig. 4.2 then integrates the media in a single MP2T stream for synchronised play-out. Fig. 4.3 describes the server/client threads in the prototype whereas Fig. 4.4 outlines the MediaSync module in greater detail.

4.3.0.1 Server-side Threads

As shown above in Fig 4.3 there is one MP2T and one MP3 streamer built on top of the Columbia University jibRTP Library. It is important to note that the jibRTP library is a bare-bones RTP and RTCP implementation. It was necessary to customise this for transport of MP2T and MP3, both of which have a nominal 90kHz clock rate. In each case, the RTP timestamp relates to the first byte of payload. For MP2T, this involves mapping between PCR and RTP, following recommended standards. For MP3, in the prototype, the frame size is 417 or 418 bytes, with a bitrate of 128kbps, thus the RTP increment between packets is the

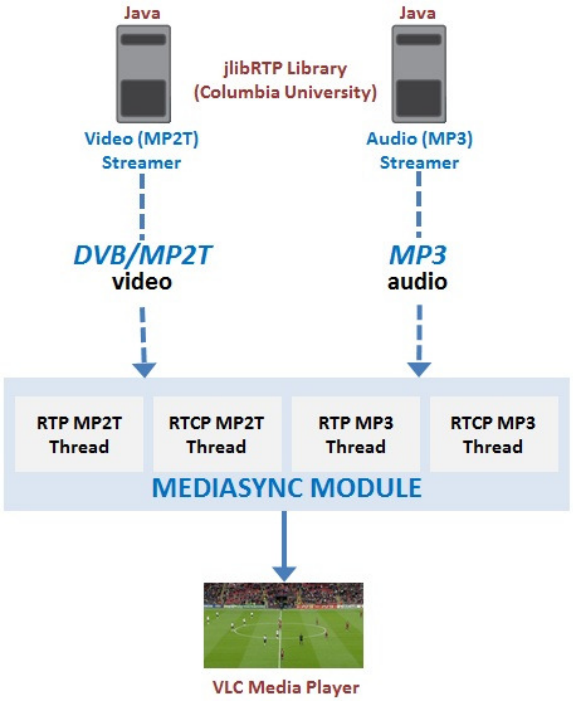


Figure 4.3: High Level Java prototype. Threads, client and media player

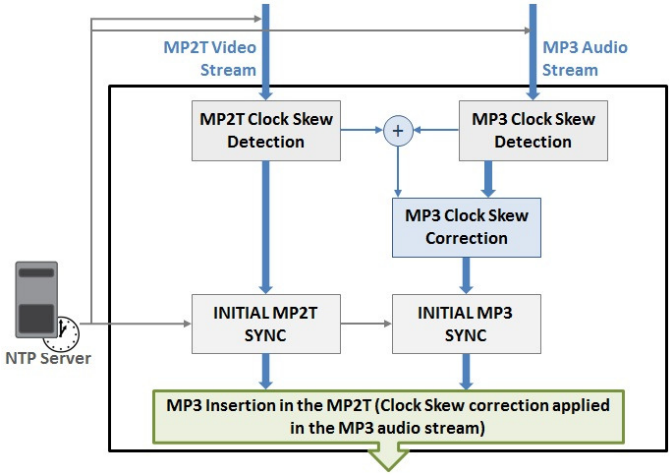


Figure 4.4: High Level description of the MediaSync Module

equivalent of 25.8125/25.875ms.

The MP2T streamer allows the user to choose the number of MP2T packets conveyed in one RTP packet. It is advised to have between 1 and 7 MP2T packets in one RTP packet (In all thesis testing, seven MP2T packets are conveyed within the RTP payload) [87].

The MP3 streamer also allows the user to choose the number of MP3 frames in one RTP

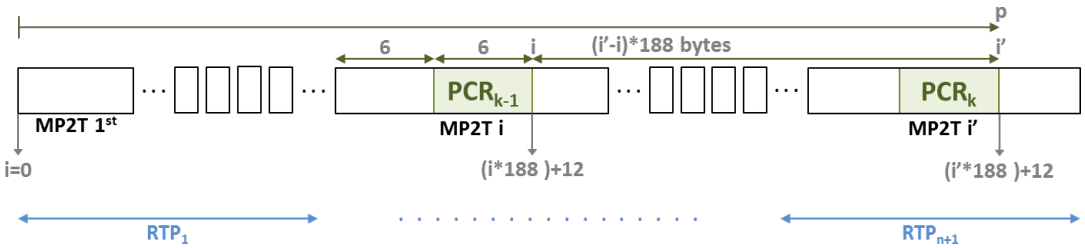


Figure 4.5: High Level diagram showing relationship between RTP and PCR in [8]

packet, although no recommendation has been found regarding this technical decision. The MP3 Streamer can not send more than two MP3 frames in one RTP packet due to the RTP packet size limit established by the RTP library used for streaming. All test cases have thus been performed with one MP3 frame in each RTP payload.

The use of RTP payload as specified in RFC 2250 for MPEG implies that the timestamps in the RTP packets conveys the media sampling of the first RTP payload byte as explained in Section 2.4.3.1 in Table 2.27 [48].

To stream MP3 audio files, RFC3119 [112] could be followed. However, the prototype does not follow this standard and instead utilises the RTP payload format for MPEG-1/MPEG-2 [48] because a more loss-tolerant RTP payload for MP3 in the prototype is out of the scope of this work.

RTP Encapsulation for MP2T The prototype implements the RTP timestamping following the time recovery system presented at ETSI TR 102 034 [8]. It is depicted in Section 3.13 in Chapter 3. The prototype at server-side applies this technique to timestamp the RTP packets based on the PCR values of the MP2T packets following packet distribution found in Fig 4.5.

The technique is based on the two clocks present at server side, the MP2T video encoder's clock and the RTP packetiser clock (synced to an NTP server for RTCP packet NTPtimestamps).

Firstly, the equation [30] is applied and that gives the Transport Rate (equation previously analysed in equation 3.21 in Chapter 3, Section 3.6.4):

$$R(i) = \frac{(i' - i'') \cdot 27MHz}{PCR(K) - PCR(K-1)} \tag{4.1}$$

Based on the value of the transport rate, the RTP timestamp can be derived based on the equation [8] (previously analysed in equation 3.23 in Chapter 3, Section 3.6.4) is:

$$PCR \cong RTP(n) + 90KHz \cdot \frac{(P+1)}{R(i)} \tag{4.2}$$

4.3.0.2 Client-side Threads

There are four client-side threads in total, two for RTP and two for RTCP. The first one RTP MP2T client-side thread that receives the MP2T packets, extracts the data and stores the MP2T packet in the MP2T buffer. The second RTP MP3 client-side thread receives the MP3 frames and extracts the data and stores the MP3 frames in the MP3 buffer. The client-side threads are depicted in Fig. 4.3.

The main client-side application runs the threads that read the MP2T and MP3 streams, and then the main application (MediaSync module) synchronises and integrates the buffered media storing the resulted media stream in a new MP2T file.

There are two other client-side threads which receive the RTCP control packets from both media streams, MP2T and MP3. These threads facilitate the initial sync and skew detection/-compensation mechanisms.

4.4 Technology used

This section describes the various tools used in prototype implementation. The media player chosen is VideoLan (VLC). VLC provides a useful error message window during the play-out of the video.

The programming language used is Java and the prototype has been developed in NetBeans. As mentioned above, the Java library used is jlibRTP library, from Columbia University. Video and audio streams use this RTP streaming library as a media delivery protocol.

The tool used to transcode the media files into the chosen video/audio codecs and media container standard is ffmpegX. A transcoder was needed to obtain the desired video in MP2T format. Moreover, transcoding the same video with different audio qualities has provided very interesting data about how audio MP2T packets are distributed in a video stream. See Table 30 in Appendix G

The tools to analyse DVB information tables are DVB Inspector and DVB Analyser. To fully understand MP2T streams, it is important to know how MPEG-2 PSI and DVB-SI tables are distributed and organised. Standards only provide the theory but a real example needs to be analysed to get the overall knowledge of the DVB and MPEG-2 systems.

The tool to analyse the MP2T packets is MPEG-2 TS Packet Analyser. This is a tool that analyses each 188 bytes packet within a Transport Stream. It gives information about the MP2T header, adaptation field and PES header. Information about the video and audio packets and, DVB-SI and MPEG-2 PSI tables is shown, although the content of the tables is not analysed. To visualise this information, the previously mentioned DVB analyser is used.

The tools to analyse and edit the MP2T video files are Smart Cutter and Avidemux. These tools also provided the functionality to cut video segments and single frames from an MP2T file as required for lab demonstrations, used to create the small video file around the first goal in the match as a proof of the difference how different audio press media describe and react to

Description	DVB
Video	MPEG-2 Colour System: yuv420p 6720x576 104857kbps
Audio	MP3 Sampling Frequency: 44.1kHz Stereo Bitrate: 128kb/s Constant Bit Rate (CBR) Language: English
Duration	51:25

Table 4.1: Original video file transcoded to a MP2T format

the same sport event.

The tool used to analyse and edit the MP3 audio files is Encspot Basic. This tool also provided the functionality to cut audio segments and single MP3 audio frames from the MP3 file. The software tool Audacity has been used to create MP3 audio files with added clock skew.

4.5 Media files used

4.5.1 Event

In order to test the prototype, an MP2T formatted video of the Champions League Final of 28 May 2011 at 07:45pm in Wembley (London), between FC Barcelona and Manchester United, is used.

4.5.2 Video

IPTV channels follow DVB-IPTV standard to broadcast their channels/programs. Transcoding the video file to MP2T has been performed with the tool *ffmepgX*. The audio characteristics are set to be equal to the Internet Radio MP3 audio file selected for testing to ease implementation complexity. The characteristics of the MP2T file are specified in Table 4.1.

4.5.3 Audio

The Internet Radio audio file of the match is from *Catalunya Radio*, the Catalan National Radio Station. The file was downloaded from the official web-page in MP3 format. The language used is Catalan. The characteristics of the MP3 file are specified in Table 4.2.

MP3 Audio	mp3 Sampling Frequency: 44.1kHz Stereo Bitrate: 128kb/s Constant Bit Rate (CBR) Duration: 05:45:52 Source: Catalunya Radio Language: Catalan

Table 4.2: Original audio file MP3 format from *Catalunya Radio* (Catalonian Radio National Station)

4.6 Solution Design

4.6.1 Audio Channel Substitution

This approach replaces the audio embedded within IPTV video with the audio from the Internet Radio service. It has certain advantages and disadvantages, as follows:

Advantages:

- SDT, PAT and PMT are identical to the original video
- MPEG-2 PSI tables are directly copied to the new MP2T stream
- DVB-SI tables are directly copied to the new MP2T stream
- Maintains the number of MP2T packets in the video stream as the original media stream because it only replaces the MP2T audio PES payload with the new audio data.

Drawbacks:

- The original audio is lost
- Only one audio channel is present in video
- User cannot change from one audio to another during the play-out

For this approach, the prototype reads the MP2T packets and if the PID equals the embedded audio channel (PID=257) it then replaces the audio content with the relevant bytes from the MP3 buffer. As outlined, this version of the prototype substitutes the original audio packets using audio packets with the same characteristics, in this case a stereo MP3 audio file at bitrate 128kbps and sampling frequency 44.1kHz.

As the MP2T packet distribution within the stream follows the same pattern, the new inserted audio packets have an identical MP2T Header as the original audio MP2T packets, and thus, PTS values are unchanged.

No further testing has been applied with this approach because the audio addition approach is considered more appealing to users, and more complex to implement.

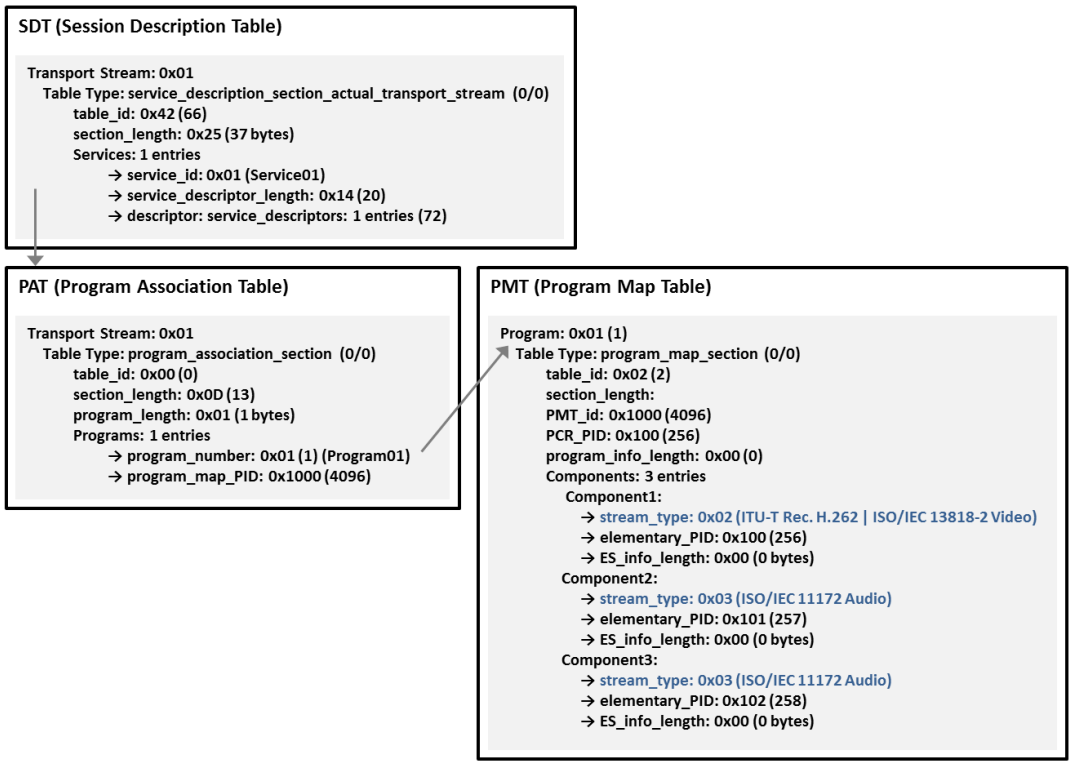


Figure 4.6: High Level DVB table structure of the prototype. In blue the video and two audio streams definitions

4.6.2 Audio Channel Addition

This approach adds an extra audio channel to the video channel from the Internet Radio channel. It has certain advantages and disadvantages, as follows:

Advantages:

- PAT and SDT are the same as original video
- The original audio stream is kept
- User can change from one audio to another during the video play-out

Drawbacks:

- The PMT needs to be modified adding an extra audio channel
- Number of audio MP2T packets is doubled

The first step is to modify the PMT table by adding the second audio stream information and assigning the PID=258 to the new audio channel. See Fig. 4.6, PMT Component 3. No other tables need to be modified.

In Table 13 in Appendix D, the new PMT table needed to describe two audio streams is shown. The prototype reads the MP2T packets and if the PID equals the audio channel (PID=257) then an extra MP2T audio packet is included (PID=258) with relevant bytes from the MP3 buffer. The final audio stream will thus have double the number of audio MP2T packets. Moreover, every time a MP2T packet with a PMT table is found, the packet is replaced with the modified PMT that includes the updated information with the second audio channel added. All DVB-SI and MPEG-2 PSI tables used in the prototype are shown in Fig. 4.6.

The audio from the Internet Radio has the same characteristics as the audio in the MP2T stream and, therefore, the MP2T Header of the new audio packets is copied from the original audio packets. As audio streams have the same characteristics, there is no need to recalculate new PTS values.

4.7 Media Delivery Protocols

4.7.1 IPTV Video Streaming

The application protocols used for the media delivery is RTP over UDP. The RTP payload used is the one defined in RFC 2250 [48]. The use of RTP is recommended when media delivery is used in IPTV media platform, but it is not compulsory. However, in this case, it is the appropriate protocol for real-time media delivery.

The specification [87] indicates that it is recommended to convey between one and seven MP2T packets within a RTP packets. As described earlier, the prototype uses in all test cases the maximum number of seven MP2T packets within every RTP packet.

4.7.2 Internet Radio Audio Streaming

The protocol used for the MP3 audio stream is also RTP over UDP using the RTP payload as defined in RFC 2250 [48]. This is appropriate as a proof of concept as the intention is to sync a radio stream delivered via IPTV. A further development of the prototype would be to use a potential HbbTV platform, thus allowing the user to select from a wider audio selection from an Internet Radio Channel. This media delivery uses and only approves HTTP Adaptive Streaming over TCP, the Standard approved by HbbNext is MPEG-DASH [113].

4.8 Bootstrapping. Sport Event Initial Information

The event's bootstrapping is done via a DVB Table, EIT, which indicates the sport event, the time and the date. The EIT table is found in Table 16 in Appendix D and represents a "present/following" EIT table of the actual MP2T stream. This table is sent at the beginning of the MP2T stream just after the general information tables SDT and PAT together with the

time-related tables TDT and TOT.

The EIT table shall be sent at least every 2s and at a minimum interval of 25ms. In the prototype, only one event is included and only one EIT table is used as a bootstrapping of the event to be synchronised, therefore, the EIT table is only sent at the beginning of streaming.

In the EIT Table 16 (Appendix D) field *start_time* lists"25/05/2011 19:45:00" and *duration* is"02:00:00". Two hours is chosen because a football game duration is 90 minutes with an added 15 minute break. This time is used to specify an agreed moment in time to initially sync MP2T and MP3 streams via NTP values. This time is used not for the actual sync but only to indicate roughly when to start the process of embedding/substituting, the precise sync is done via NTP/PCR/RTP as described later.

The EIT table has two descriptors, *content_descriptor* and *short_event_descriptor*. The former indicates the program category of the event, in the prototype "sports", in field *content_nibble_level_1* and "football" in field *content_nibble_level_2* . The latter descriptor gives information about the language used in field *ISO_639_language_code* value "eng", the event name, *event_name* "ChampionsLeague2011" and descriptive text, *text_char* "Barca vs ManU".

4.9 Initial Sync

The Initial Sync prototype is divided in two main parts, the MP2T stream Initial Sync and the MP3 stream Initial Sync. Both use RTP timestamps and wall-clock time (NTP) taken from RTCP to indicate the beginning of the sport event. As described in the previous Section 2.4.1, the RTP encapsulation of both the MP2T and MP3, as well as the generation of associated RTCP streams is required to facilitate sync and subsequent skew detection/compensation. The MP2T Initial Sync is further based on the RTP timestamp (with mapping to wall-clock NTP) and the PCR values within the MP2T stream, whereas the MP3 Initial sync is based on the RTP timestamps (with mapping to wall-clock NTP) and the MP3 frame equivalent in time values.

The prototype uses TDT and TOT tables at beginning of the MP2T streams to transmit the IPTV time values to the client plus the information about the beginning of the event is conveyed within the EIT Table shown in Table 16. The information within the tables can be found in TDT Table 17 and TOT Table 18 in Appendix D.

The beginning of the game in UTC time from the EIT is simply used as an agreed moment in time when the MP2T and MP3 stream initially sync. This variable is known as $\text{MP2T}_{\text{ntpStart}}$ in Fig. 4.7. As granularity of this time is seconds, it is important to clarify that this time is not used for precise synchronisation but only to indicate when to begin the synchronisation.

A different scenario to consider is if user requires the sync after the sport event has began and the EIT time has already passed. Thus, process would simply require an agreement at time T on when to start synchronisation e.g., T+10s.

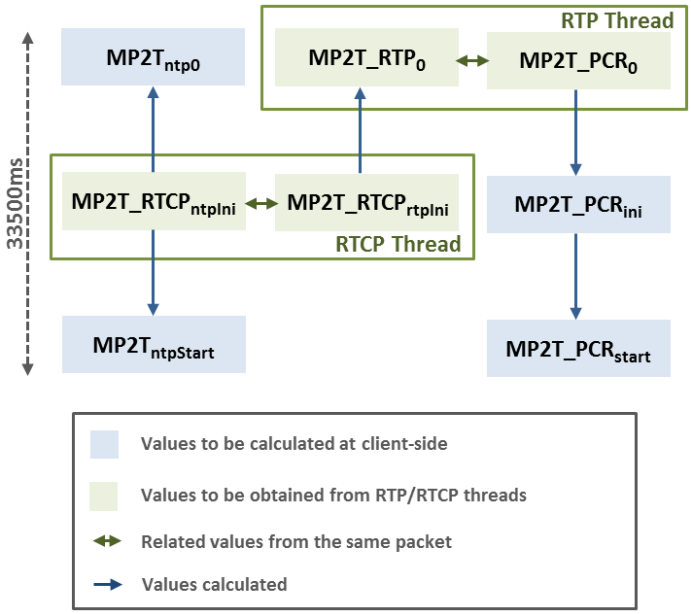


Figure 4.7: Initial Sync performed in the MP2T video stream at client-side. Terms found in Table 4.3

Value	Moment	Description
NTP	$MP2T_{ntp0}$	Derived wall-clock time related to 1st RTP packet
	$MP2T_RTCP_{ntpIni}$	Wall-clock time from 1st RTCP SR packet
	$MP2T_{ntpStart}$	Wall-clock time representing advertised beginning of sport event (second level granularity)
RTP	$MP2T_RTP_0$	RTP timestamp from 1st RTP packet
	$MP2T_RTCP_{rtpIni}$	RTP timestamp from 1st RTCP packet
MP2T PCR	$MP2T_PCR_0$	PCR value 1st RTP-MP2T packet
	$MP2T_PCR_{ini}$	Derived PCR value 1st RTCP-MP2T packet
	$MP2T_PCR_{start}$	Derived PCR value representing advertised beginning of sport event

Table 4.3: Description of symbols used in Fig. 4.7

4.9.1 MP2T Work-flow

The work-flow for Initial Sync within the MP2T stream is shown in Fig. 4.8 The first step is performed when the first RTCP packets are received at client-side. This RTCP NTP value is called $MP2T_RTCP_{ntpIni}$. Recall that in the present scenario, the synchronisation is automatically started based on data in EIT table about the kick-off time with a granularity of a second, referred to here as $MP2T_{ntpStart}$. The EIT values are listed in Table 16 in Appendix D.

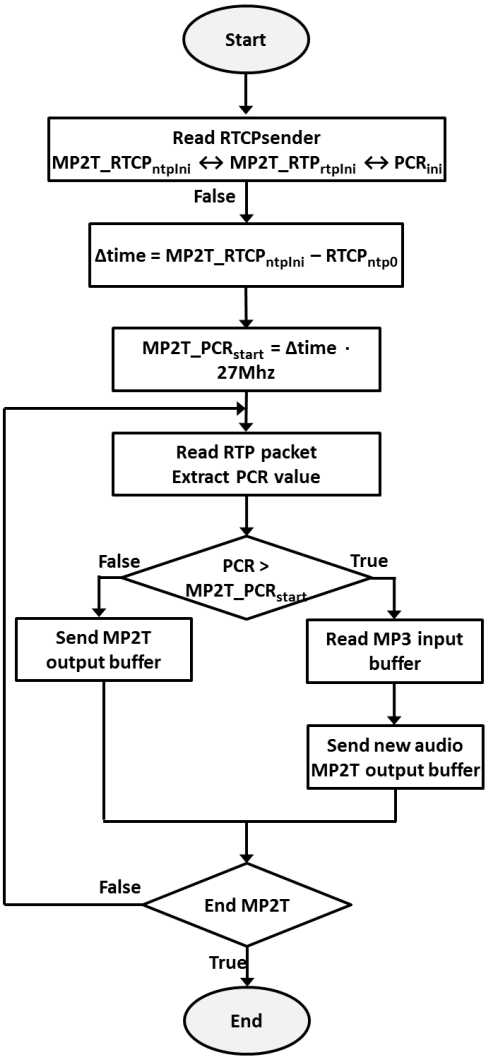


Figure 4.8: Initial Sync performed in the MP2T video stream at client-side. Terms found in Table 4.3

$$\begin{cases} MP2T_{ntpStart} = 1357415100 \leftrightarrow 25/5/2011\ \ 19:45:00.000 \\ MP2T_{ntp0} \quad\ = 1357414765 \leftrightarrow 25/5/2011\ \ 19:39:25.000 \end{cases} \tag{4.3}$$

From the first RTCP packet received, the values MP2T_RTCP$_{\text{ntpIni}}$ and MP2T_RTCP$_{\text{rtpIni}}$ are stored. After the first RTCP is received the prototype can relate all RTP packet timestamps back to wall-clock time and, in particular, the first one, named here as MP2T$_{\text{ntp0}}$, i.e., MP2T_RTP$_0$ is mapped back to its equivalent NTP time.

The equivalent in time of PCRs values is straight forward considering that the PCR clock

runs at 27MHz. In the video sample used in the prototype the sport event advertised kick-off is at time 05:35s (335s) after the wall-clock time relating to when first RTP packet is received, $\text{MP2T}_{\text{ntp0}}$. Thus, the sport event advertised start time will relate to an increment in PCR equivalent to 335000ms.

$$\Delta Time = MP2T_{ntpStart} - MP2T_RTCP_{ntpIni} = 335000ms \tag{4.4}$$

The PCR equivalent to this time difference needs to be found to calculate when the audio insertion (either addition or substitution) in the MP2T stream should commence. This instant is shown in Fig. 4.7 as $\text{MP2T_PCR}_{\text{start}}$, and represents the time in PCR terms equivalent to the wall clock time of $\text{MP2T}_{\text{_ntpStart}}$.

In Fig. 4.7 relationship between all the RTP, NTP and PCR values and their source is visualised for the MP2T Initial Sync process whereas in Table 4.3, the meaning of the variables used is explained. Fig. 4.8 outlines the flowchart for this process. To summarise, the process consist of two stages. First, when the first RTP packet, containing PCR values, arrives at client. Second, when the first RTCP SR packet also arrives at the receiver. Those two steps contain the information needed for the MP2T Initial Sync.

The first stage commences when the first RTP packets arrive, with a PCR value, the prototype stores MP2T_RTP_0, MP2T_PCR_0. The second stage, when the first RTCP packet is received the prototype stores $\text{MP2T}_{\text{ntpIni}}$ and $\text{MP2T_RTCP}_{\text{rtpIni}}$.

At this stage, the process has the values $\text{MP2T}_{\text{ntpIni}}$, $\text{MP2T_RTCP}_{\text{rtpIni}}$ from the MP2T RTCP thread and MP2T_RTP_0 and MP2T_PCR_0 from the RTP thread. The variable $\text{MP2T}_{\text{ntp0}}$ is then derived by determining the difference in RTP between $\text{MP2T_RTCP}_{\text{rtpIni}}$ and MP2T_RTP_0, and translating this to wall-clock time. Finally, knowing MP2T_PCR_0, the prototype obtains the value of $\text{MP2T_PCR}_{\text{start}}$ which is the time in PCR terms of the advertised sport event $\text{MP2T}_{\text{ntpStart}}$ used for the MP2T stream initial sync.

$$\begin{cases} MP2T_{ntp0} = MP2T_RTCP_{ntpIni} - (MP2T_RTCP_{rtpIni} - MP2T_{rtp0}) \\ MP2T_{ntpStart} = MP2T_{ntp0} + 335000 \\ MP2T_{pcrIni} = ((MP2T_RTCP_{ntpIni} - MP2T_{ntp0}) * 27000) + MP2T_{pcr0} \\ MP2T_{pcrStart} = ((MP2T_{ntpStart} - MP2T_{ntpIni}) * 27000) + MP2T_{pcrIni} \end{cases} \tag{4.5}$$

4.9.2 MP3 Work-flow

The MP3 Initial Sync is similarly based on the information collected when the first RTP and RTCP packets are received. When the first RTCP SR packet is received for MP3 stream, the prototype extracts and stores the NTP and RTP timestamps. Fig. 4.9 depicts the relationship between all the RTP and NTP values and their source for the MP3 Initial Sync and as with MP2T, Table 4.4 describes the meaning of the variables used. Fig. 4.10 illustrates the flowchart

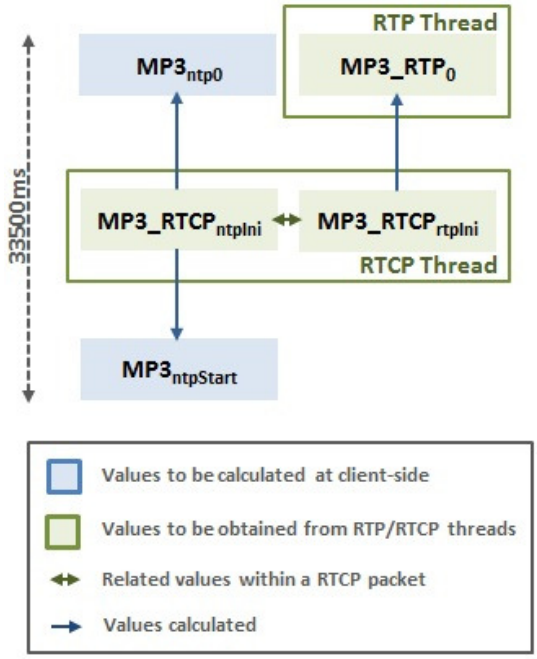


Figure 4.9: Initial Sync performed in the MP3 video stream at client-side. Terms found in Table 4.4

Value	Moment	Description
NTP	$MP3_{ntp0}$	Derived wall-clock time 1st RTP packet
	$MP3_RTCP_{ntpIni}$	Wall-clock time from 1st RTCP SR packet
	$MP3_{ntpStart}$	Wall-clock time advertising beginning of sport event
RTP	$MP3_RTP_0$	RTP timestamp 1st RTP packet
	$MP3_RTCP_{rtpIni}$	RTP timestamp 1st RTCP packet

Table 4.4: Description of Symbols used for MP3 in Fig. 4.9

of the mechanism.

When the MP3 RTP thread receives an RTP packet at the client, it analyses the MP3 frame in the RTP payload and its time value by means of equation 4.3 based on the MPEG Audio Layer. This is used by the prototype to estimate the elapsed time.

Identical to the MP2T Initial Sync process, the MP3 Initial Sync has two steps. The first is to extract information when the first MP3 RTP packet arrives and second when the first MP3 RTCP SR packet is received at the client-side. As such, when the first RTP packets arrives, the value of the first RTP timestamp is extracted and stored as $MP3_RTP_0$, when the first RTCP packet arrives the prototype extracts and stores $MP3_RTCP_{ntpIni}$ and $MP3_RTCP_{rtpIni}$.

Knowing $MP3_RTCP_{ntpIni}$ and $MP3_RTCP_{rtpIni}$ from the RTCP Thread and $MP3_RTP_0$, the value of $MP3_{ntp0}$ is obtained. Finally, the difference between $MP2T_{ntpStart}$ i.e., from the MP2T EIT table and $MP3_{ntp0}$ gives the time remaining to the advertised kick off of the game.

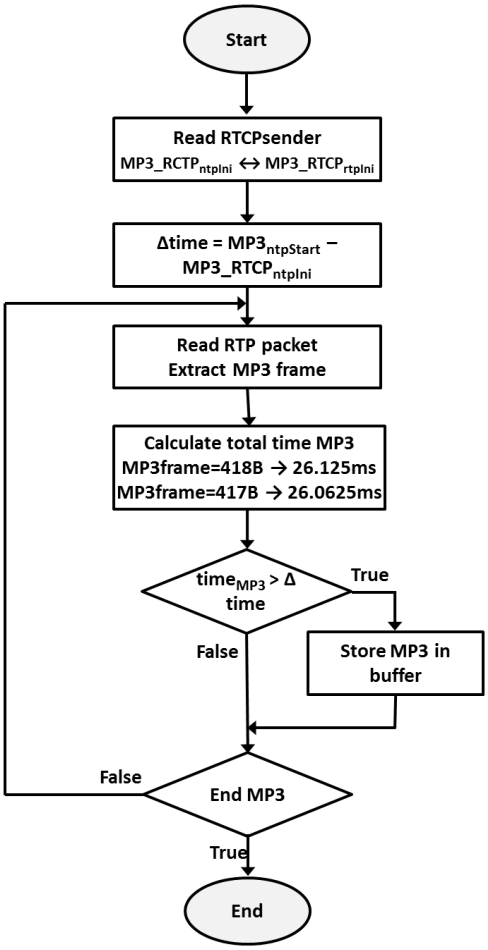


Figure 4.10: Initial Sync performed in the MP3 audio stream at client-side.Terms found in Table 4.4

The time equivalent is calculated every time an MP3 frame is received by the client and the value of Time_{MP3} is incremented. When Time_{MP3} reaches the $\text{MP2T}_{\text{ntpStart}}$, then the MP3 audio frames are stored in the audio buffer, ready for addition/submission.

$$\begin{cases} MP3_{ntp0} = MP3_RTCP_{ntpIni} - (MP3_RTCP_{rtpIni} - MP3_{rtp0}) \\ MP3_{ntpStart} = MP3_{ntp0} + 335000 \end{cases} \tag{4.6}$$

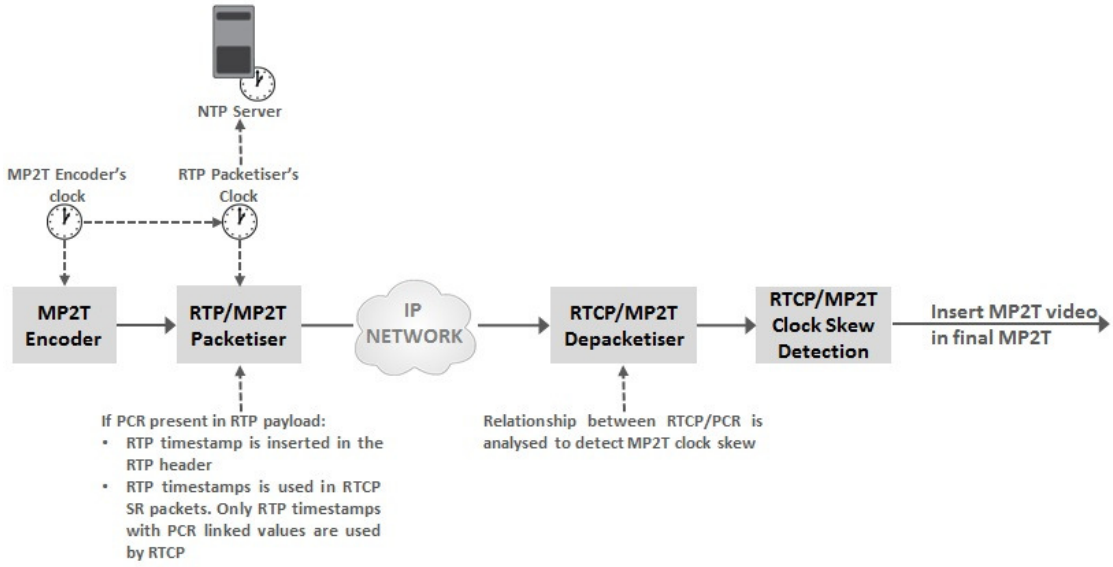


Figure 4.11: MP2T Encoder's and RTP packetiser clocks

4.10 MP2T Clock Skew Detection

The RTP timestamps are inserted in the MP2T server-side as explained in 4.3.0.1 following the time recovery system presented in ETSI TS 102 034 [8]. The main challenge in applying this formula is how to calculate the RTP timestamp for RTP packets which don't convey any PCR value in the MP2T packets within the RTP payload. The solution applied is to apply the formula when possible and for those packets without PCR values, an average of an increment of 2.1ms[1] is applied to achieve the correct time for the MP2T video file to be streamed.

The high level MP2T streaming and RTP packetising, with the related clock relations to NTP server, is shown in Fig. 4.11. The MP2T media encoder has its internal clock, and also the RTP/MP2T packetiser clock is related to the NTP wall-clock time, synchronised via the NTP server. After streaming across the network via RTP, the media packets are depacketised and stored in the receiver prior to audio insertion and play-out. The MP2T clock skew detection method will be triggered once RTCP SR packets are received. Fig. 4.11 shows the timing link between encoder and media server clock synchronised via the NTP server.

The client-side skew detection mechanism detects clock skew based on the received RTCP SR packets. Recall that the server RTCP thread process only sends packets with true RTP timestamps of RTP if an encapsulated MP2T packet has a PCR value. RTCP SR packets provide mapping between a RTP/PCR values and an NTP value to detect the encoder clock

[1] Value chose as the closest value matching video file duration with streaming time

skew:

$$ClockSkew_{\text{MP2T}} = \frac{NTP_n - NTP_{n-1}}{PCR_n - PCR_{n-1}} \begin{cases} > 1 & \text{Clock skew positive} \\ = 1 & \text{No Clock skew} \\ < 1 & \text{Clock skew negative} \end{cases} \quad (4.7)$$

Note that clock skew detection based on ETSI 102 034 did not work so a workaround was developed as explained later. The MP2T clock skew is achieved by calculating the clock skew average of all RTCP SR packets analysed.

Further implementation details of this process, involving steps at both server and client are as follows: On the server-side a global scope class stores the most recent RTP and PCR values each time an RTP packet is generated.

When the server RTCP thread wishes to create/send an RTCP packet (typically 5s), it populates the RTP and NTP timestamp fields using the above class values.

On the receiver (client) side, the RTP receive thread stores the PCR and RTP values in an *arrayList* data structure. When the RTCP receive thread receives an RTCP SR packet from server, it extracts the RTP and NTP timestamps. A corresponding RTP timestamp is searched for in the *arrayList* and the associated PCR value is retrieved, which gives the final relationship between the RTP timestamp and a PCR value and the NTP value associated to the RTP timestamp. This PCR value, related to an NTP value is used as above to detect MP2T clock skew. The difference between two consecutive NTP values is compared (NTP_n NTP_{n-1}) with the difference between two consecutive PCR values, PCR_n-PCR_{n-1}. In equation 4.7 the Clock Skew values are described - a value > 1 represents positive clock skew, < 1 represents negative clock skew and if ratio is 1, then no clock skew is detected.

On the client-side, the Flow Chart for analysing the MP2T clock skew is presented in Fig. 4.12. Essentially, every time an RTCP SR is received at client-side, MP2T clock skew is calculated.

4.11 MP3 Clock Skew Detection and Correction

A range of MP3 clock skew detection and correction techniques are proposed in this section. Regarding detection, two techniques are described. The first technique follows the fundamentals outlined ETSI TS 102 034 [8] and is also used for clock skew detection for MP2T streams, as described previous section on MP2T skew. The second uses RTP timestamps only by mapping RTP to wall-clock time.

As described earlier, MP3 audio files don't carry clock references or timestamps. Therefore, the MP3 audio file is adapted by using the audio bitrate (in this case 128kbps) to detect and correct clock skew in MP3 audio files.

As described, the prototype either inserts an added audio stream or substitutes an audio stream into the final MP2T stream. Prior to this final step, the prototype applies MP3 clock skew detection and further correction if needed to the MP3 audio file. Thus, clock skew issues

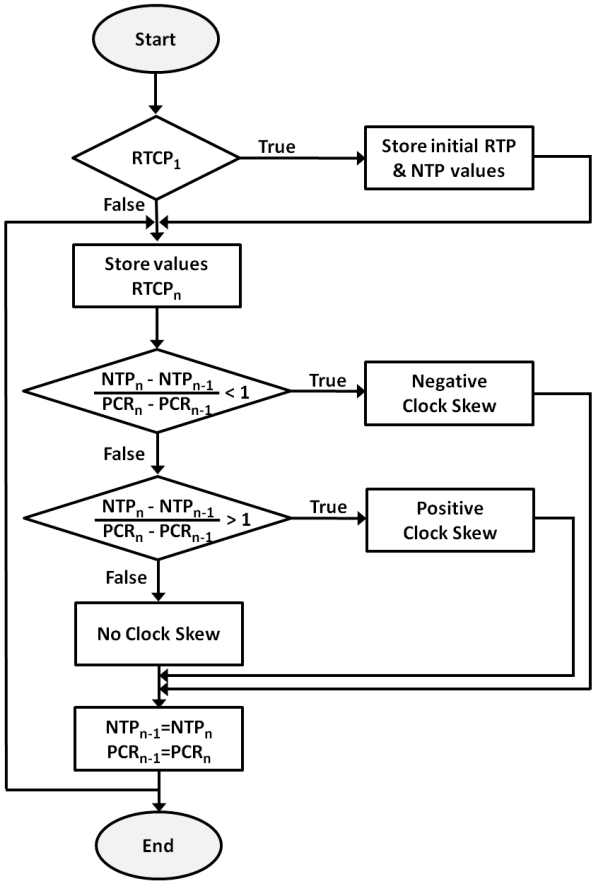


Figure 4.12: Flowchart MP2T Clock Skew detection mechanism

are resolved before packets are multiplexed to the MP2T stream.

Regardless of skew detection method, the prototype applies the mechanism every second. If clock skew is detected then the correction technique is applied, as described later.

The techniques are based on the two clocks present on the server side, the MP3 audio encoder's clock and the RTP packetiser clock although the role of the latter differs between techniques. In Fig. 4.13 all clocks involved in the solution are shown. In Fig. 4.14 the general work-flow of the techniques (skew detection and correction) is illustrated.

The first skew detection method assumes that RTP is tied to a wall-clock rather than related to media rate or number of bytes. The second is based on the premise that the RTP timestamp is mapped directly to media rate, similar to VoIP applications, and thus RTCP is used to detect clock skew as it maps RTP timestamp to wall-clock NTP values. These methods are outlined in detail in the next sections.

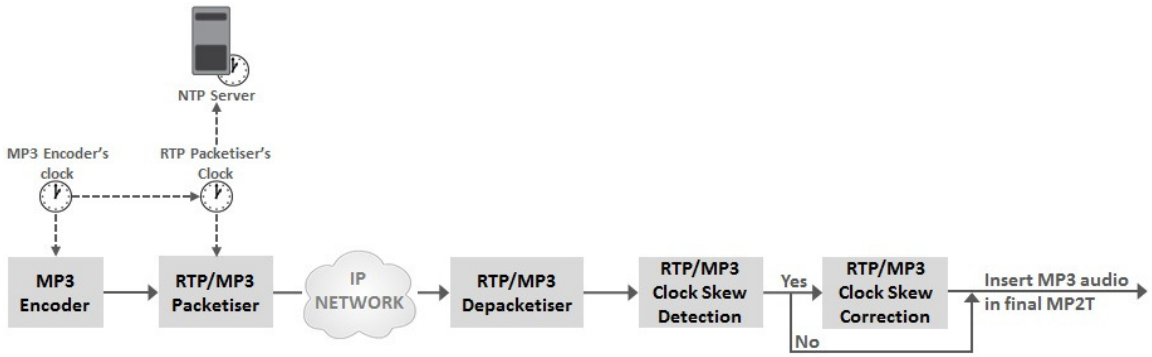


Figure 4.13: MP3 Encoder's and RTP packetiser clocks

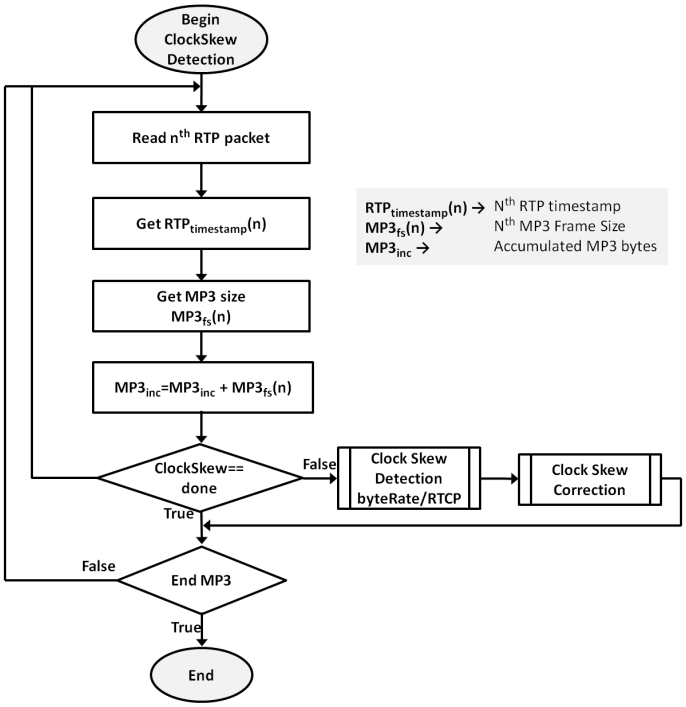


Figure 4.14: Common MP3 Clock Skew Correction Technique for the two MP3 Clock Skew detection techniques applied

4.11.1 MP3 Clock Skew Detection

The sample media file has MP3 frames size of 418 or 417 bytes. As described earlier, every RTP packet conveys a single MP3 frame due to the RTP Library maximum RTP payload size allowed. The MP3 frame payload is, due to the 4 byte MP3 Header size, 414 bytes for the former frame size or 413 bytes for the latter. The RTP timestamps values are inserted at MP3 server-side as described in Appendix E in Table 21.

The relevant RFC to stream the MP3 audio stream is RFC 2250 [48] which establishes the meaning of the RTP timestamp value as *'timestamp: 32 bit 90k Hz timestamp representing the target transmission time for the first byte of the packet payload'*. This payload is especially relevant when clock skew detection is applied because in the two possible methods used, the RTP timestamp increments in order to compare with the number of bits in one case, and with the NTP increment in another.

4.11.1.1 Clock Skew Detection by Means of MP3 Frame Size

The key point of this procedure is to compare the wall-clock time taken to sample the number of bytes of an MP3 frame. If MP3 frame size is 417 bytes (413 bytes MP3 payload) that means, using our media sampling rate value 128kbps, it has 25.8125ms time equivalent data. Whereas, an MP3 frame size of 418 bytes (414 bytes MP3 payload) represents 25.875ms time value.

Attempting to detect clock skew on a per frame basis is not feasible due to the very short elapsed time and typical clock skews. For example, a clock skew of 100ppm is typical of consumer grade quartz crystals. If clock skew is exaggerated to say 1600ppm, then the following analysis illustrates the challenge.of detecting clock skew after every MP3 frame. For an MP3 frame size of 417/418 bytes the clock skew offset arising from this would be:

$$417 MP3 Framesize \rightarrow \frac{25.8125 \cdot 1.6}{1000} = 0.0413 ms \tag{4.8}$$

$$418 MP3 Framesize \rightarrow \frac{25.875 \cdot 1.6}{1000} = 0.0414 ms \tag{4.9}$$

As previously calculated that means:

$$417 \rightarrow MP3 Framesize \rightarrow RTP_{timestamp} = 0.0413 ms \cdot 90k = 3.717 \tag{4.10}$$

$$418 \rightarrow MP3 Framesize \rightarrow RTP_{timestamp} = 0.0414 ms \cdot 90k = 3.7 \tag{4.11}$$

Therefore, detecting much lower values of clock skew at MP3 frame level and applying clock skew correction is not feasible due to the small values that need to be considered. Such small clock skew level would require correcting the clock skew by adding/removing a specific number of bits instead of a single byte and that is not possible with the MP3 frame structure which only allows MP3 frames with a whole byte number size.

A more practical solution is to detect clock skew on a per second basis and to correct it by adding/removing an entire byte or MP3 frame, as described in subsequent sections.

4.11.1.2 Method 1: Clock Skew detection by means of Sampling Bit Rate via RTP with latter derived from wall-clock time

Every RTP packet contains a single MP3 frame, thus, when the packet arrives the total number of bytes received is incremented with the MP3 frame size. When the audio bit rate for one second is reached, i.e., 128kb, the difference is RTP timestamp values is determined, $\Delta RTP_{tms}(x)$. If the difference is not 1s (a RTP timestamp increment of 90k) then clock skew is detected, positive and negative. In the event of clock skew the MP3 clock skew mechanism will be applied. In Fig. 4.15a a the high level work-flow of the clock skew detection technique is illustrated, and in Fig. 4.14 and in Fig. 4.17 the correction level flow chart is presented.

Fig. 4.15a shows the work-flow for the clock skew detection mechanism. Fig. 4.14 outlines the general work-flow which shows that every time an RTP packet is received, the number of MP3 bytes is counted (since the last clock skew correction has taken place). Subsequently, the clock skew detection function runs and if the number of bytes is bigger than 128k, the correction method takes place.

The flowchart for setting the clock skew level to be applied to the clock skew correction if found in Fig. 4.16. This step occurs prior to the MP3 clock skew correction. The prototype detects the correct clock skew level but only applies three levels related with the levels of correcting one, two or three bytes, as is explained in Section 4.11.2.

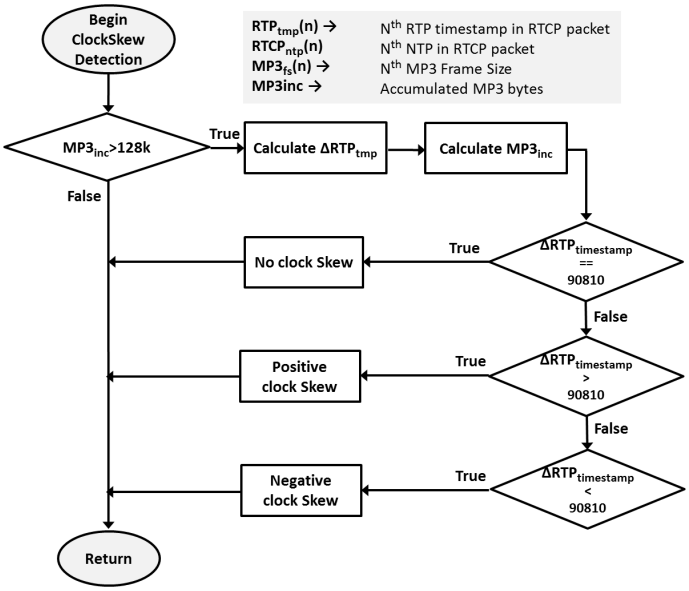
4.11.1.3 Method 2: Clock Skew detection by means of RTCP

In this approach shown in Fig. 4.15b, the RTP encapsulation of MP3 timestamp value is set by the MP3 encoder rate. The clock skew detection is performed once consecutive RTCP packets are received at client-side. RTCP values are stored and compared with the values of previous received RTCP. The increment of the RTP timestamp and the NTP value is calculated. Then ΔNTP is then divided by ΔRTPtimestamp. This value indicates the clock skew.

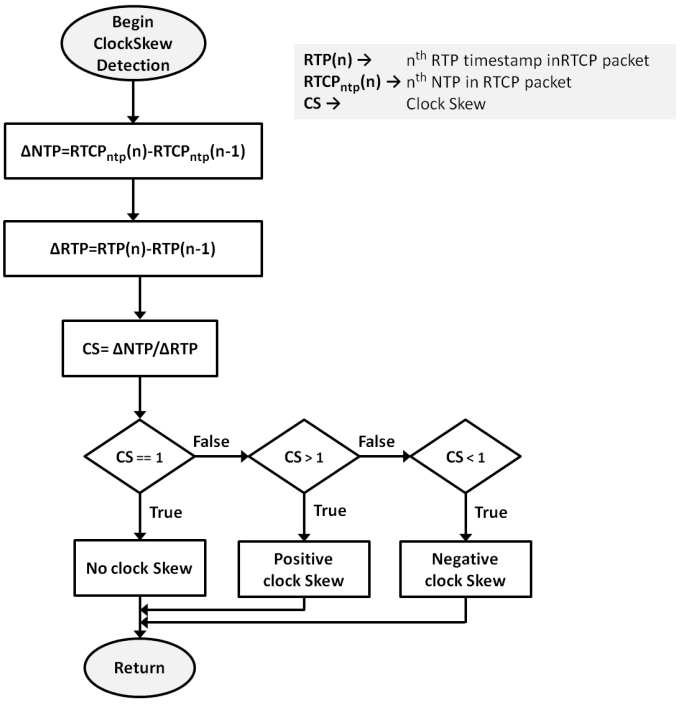
Every time a RTCP SR packet is received, it calculates the difference between the RTP timestamps values and the difference between the two consecutive NTP values relative to the previous SR packet. Clock skew is the division between ΔNTP and ΔRTP. As before, if the ratio is equal to 1 then no clock skew is detected, if ratio is > 1, then positive clock skew is detected, if ratio is < 1, negative clock skew value is detected. The clock skew level is stored for the clock skew correction mechanism.

4.11.2 MP3 Clock Correction

As described above, clock skew detection using either mechanism occurs every second if the previous clock correction has been applied. Two MP3 clock skew correction solutions have also been proposed. The first one applies the correction periodically (at fixed periods in time) and thus a variable number of bytes are modified (added/removed) depending on skew rate, which in this case happens every second. The second step applies the correction of a full MP3 frame



(a) MP3 Clock Skew Detection Work-flow via MP3 bitrate



(b) MP3 Clock Skew Detection Work-flow via RTCP

Figure 4.15: MP3 Clock Skew Detection Work-flow

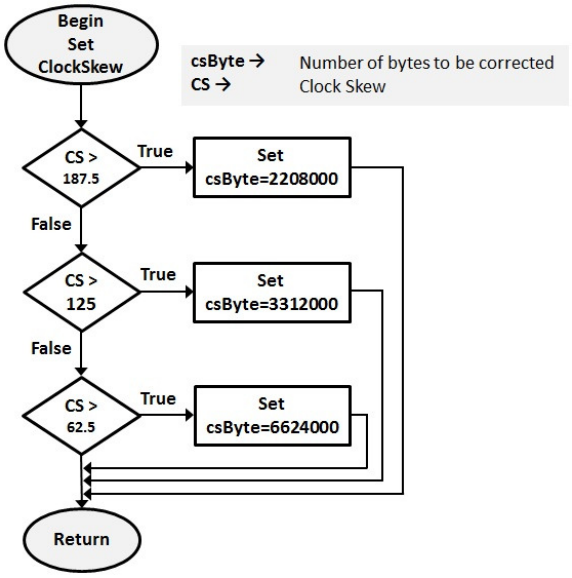


Figure 4.16: MP3 Flow Chart Clock Skew Set Level

applied over a variable period of time, so as to correctly correct for skew.

In both techniques the correction method applied is almost identical. On the one hand, if positive clock skew is detected the correction is applied by removing bytes. On the other hand, if negative clock skew is detected the correction is applied by adding stuffing bytes. The difference between the two techniques is the amount of bytes to remove/add into the MP3 audio stream, in the first case it is only one byte, while in the second case it is an entire MP3 frame.

One step needed when modifying an MP3 frame is that the MP3 Header changes (modifying *padding* field) to apply the addition or deletion of a byte within an MP3 Frame. Note that only one byte in every MP3 frame is added/removed, so that the MP3 frame structure does not get altered. Only one byte can be removed when the MP3 frame is of 418 byte size and only one byte can be added when the MP3 frame is of 417 byte size. This is fully explained in Section 4.11.2.3.

Table 4.5 shows the header when positive clock skew correction needs to be applied whereas in Table 4.6 shows the header when negative clock skew correction is applied.

4.11.2.1 Thresholds for MP3 Clock Skew Correction

The threshold levels for correction have been derived from the minimum correction available to apply within an MP3 frame. The prototype needs to maintain a correct MP3 audio file. Therefore, the size of MP3 frames need to comply to the standard as a random number of bits can't be deleted or added. First, it always needs to take into account that it needs to be an entire number of bytes, so the frame size is coherent with the standard. Second, it can only be

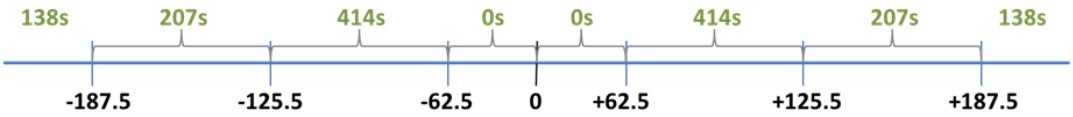


Figure 4.17: MP3 Correction thresholds applied in prototype

Original MP3 Frame Header	Size=413	0xff 0xfb 0x94 0x40
Final MP3 Frame Header	Size=414	0xff 0xfb 0x90 0x64

Table 4.5: MP3 Frame Headers modification when positive clock skew (Delete one byte to the original MP3 frame)

Original MP3 Frame Header	Size=414	0xff 0xfb 0x90 0x64
Final MP3 Frame Header	Size=413	0xff 0xfb 0x94 0x40

Table 4.6: MP3 Frame Headers modification when negative clock skew (Add one byte to the original MP3 frame)

one byte per MP3 frame because it maintain the correct audio file MP3 Header format. To fix only one byte at each frame, only the field in the MP3 frame *padding* is required to be changed to indicate the change of the MP3 frame size. Fig. 4.17 shows the thresholds applied.

The levels for clock skew corrected every second are found in Table 4.7 whereas the levels for clock skew corrected at variable frequency but at a fixed number of bytes(full frame addition/deletion) are found in Table 4.8.

At a byte level, Table 4.7, shows that 3 bytes will be added/removed if clock skew is bigger than 187.5ppm. 2 bytes are corrected if clock skew is between 125 and 187.5ppm, and finally one byte correction is applied when clock skew is between 62.5ppm and 125ppm. For clock skew smaller than 62.5 clock skew correction is not applied.

At MP3 level, Table 4.8 shows that the same levels of clock correction are used but the time interval is variable depending on the clock skew. Therefore, for clock skew greater than 187.5ppm, clock skew will be corrected after 2208000 bytes, between 125 and 187.5ppm, it occurs after 3312000 bytes, and finally between 62.5 and 125ppm clock skew correction is applied after 6624000 bytes. There is a majority of frames of size 417 bytes in the MP3 audio therefore the number of bytes is calculated by multiplying the time by the 16kbyte (128kbs) bitrate.

4.11.2.2 Correction Every Second by a Variable Number of Bytes

This solution has been implemented and three levels of correction per second can be applied; one, two and three bytes, following Table 4.7 levels. That provided a maximum of 187.5ppm clock skew correction.

This clock correction technique has to conform to the MP3 frame size limitation. This means that only in an MP3 frame size of 418 bytes can positive clock skew (delete a byte) can

Clock Skew (ppm)	Bytes	Distribution of bytes correction
ClockSkew>187.5	3	Bytes corrected first 3 MP3 frames of every second
187.5>ClockSkew>125	2	Bytes corrected first 2 MP3 frames of every second
125>ClockSkew>62.5	1	Byte corrected first MP3 frame of every second
62.5>ClockSkew>0	0	No bytes corrected

Table 4.7: Clock Skew Correction levels for fixed time intervals

Clock Skew (ppm)	Time Correction	Time Correction (s)	Bytes
ClockSkew>187.5	2min 18.0s	138.0s·16k	2208000
187.5>ClockSkew>125	3min 27s	207.0s·16k	3312000
125>ClockSkew>62.5	6min 54.0s	414.0s·16k	6624000
62.5>ClockSkew>0	0s	0s	0

Table 4.8: Clock Skew Analysis for fixed correction over adaptive time

be applied. Moreover only in a 417 byte size MP3 frame can negative clock skew (add a byte) be applied. In both cases the MP3 frame header should be updated by modifying the value of *padding* field. This is calculated by the equations 2.1 and 2.2 that give the MP3 Frame size:

$$MP3Frame_{Size} = \begin{cases} 418bytes & Positive\ clock\ skew\ \rightarrow -1\ byte \rightarrow padding = 0 \\ 417bytes & Negative\ clock\ skew \rightarrow +1\ byte \rightarrow padding = 1 \end{cases} \quad (4.12)$$

The correction technique waits until an appropriate MP3 is found. E.g., positive clock skew correction waits until a 418 MP3 frame size is found (to remove a byte) and negative clock skew correction waits until a 417 MP3 frame size is found (to add a byte).

There is a maximum of one byte per frame that can be corrected (delete in positive clock skew and add in negative clock skew). Therefore, if more than one byte needs to be corrected the correction is applied in consecutive MP3 frames, two or three based on the level, always waiting for the correct MP3 frame size.

Fig. 4.18 shows the entire byte correction applied within an MP3 Frame, whereas Fig. 4.19 shows the bits distributed within a MP3 frame.

Table 4.7 shows the clock correction levels. If clock skew (ppm) is 125ppm>clock skew>62.5ppm only one byte is corrected. If clock skew is 187.5ppm>clock skew>125ppm two bytes are corrected and finally, if clock skew>187.5ppm three bytes are corrected.

Three scenarios have been applied, adding/removing a byte at beginning of MP3 frame, after the MP3 header, or at the end. Finally, the technique to add/remove the 8-bits from an MP3 frame in a distributed way within the MP3 frame also was also tested. The results of three options were the same, i.e., sound quality degraded, as is further explained in Chapter 5.

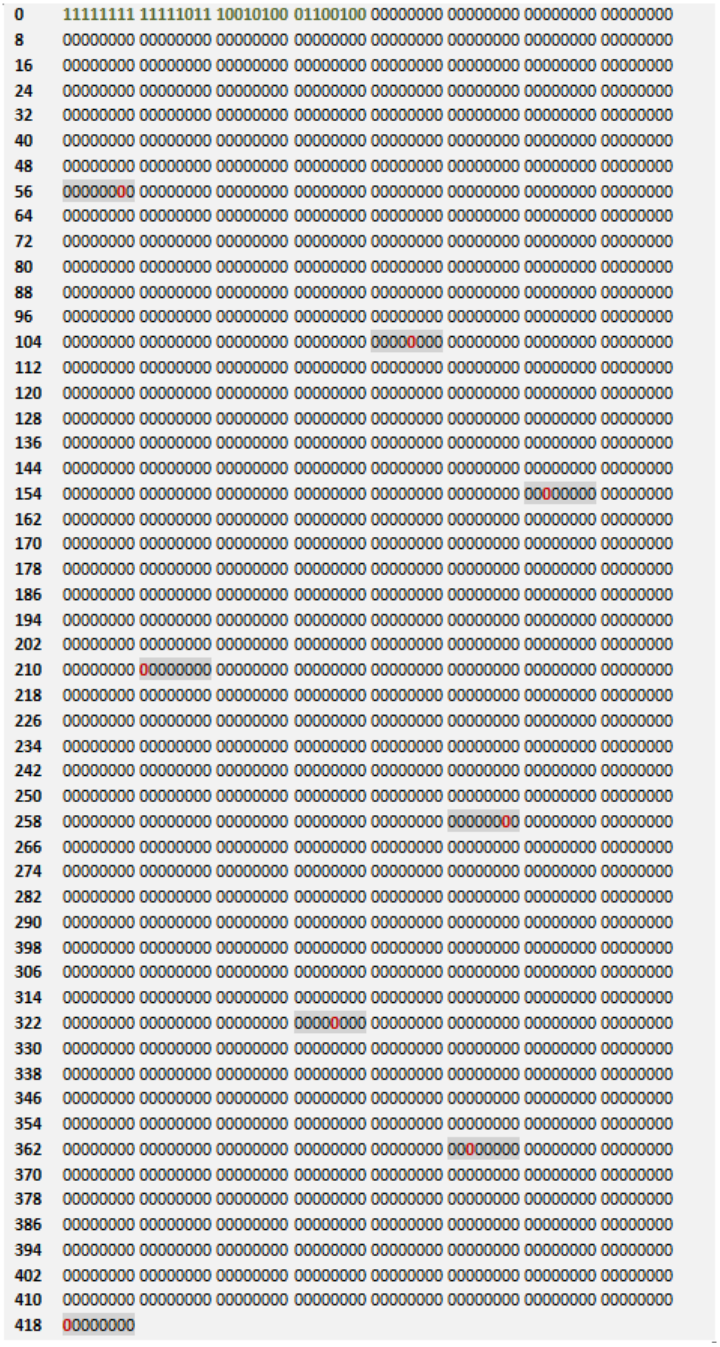


Figure 4.18: MP3 8 bits clock skew correction distributed within the MP3 Frame. The bits in green show the MP3 Frame Header. Bits coloured in red show the bits added/deleted within the frame

0	ff fb 94 40 xx xx xx xx xx xx xx xx xx xx xx xx
32	xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx
64	xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx
96	xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx
128	xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx
160	xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx
192	xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx
224	xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx
256	xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx
288	xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx
320	xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx
352	xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx
384	xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx
416	xx xx xx

Figure 4.19: MP3 entire byte correction within a MP3 Frame. The bits in green show the MP3 Frame Header the byte in red is the byte to added/deleted in the clock skew correction model

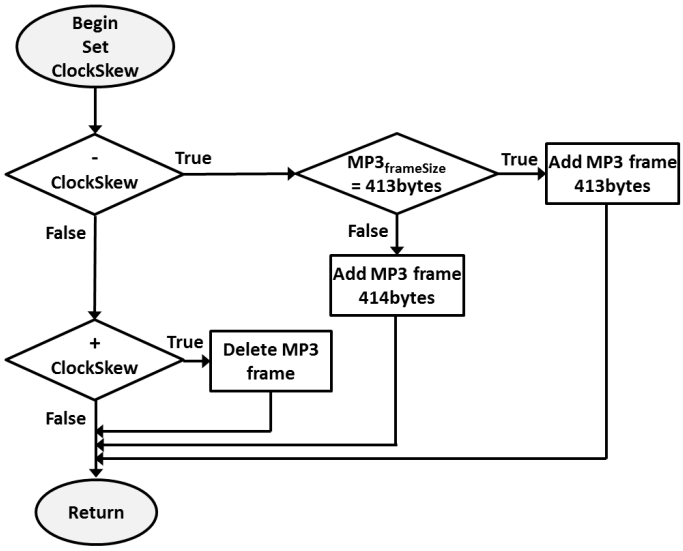


Figure 4.20: MP3 Clock Skew Correction based on a fixed MP3 frame

4.11.2.3 Correction by an MP3 Frame in Variable Time Period

This technique, as opposed to the previous one, selects a fixed number of bytes (MP3 frame size) and applies the correction techniques at the appropriate times when required.

The correction is applied to an entire MP3 frame. For positive clock skew, a full MP3 frame is deleted and for negative clock skew a stuffing MP3 frame is added. The time values of MP3 corrections are listed in Table 4.8. Fig. 4.20 shows the work-flow of the correction at MP3

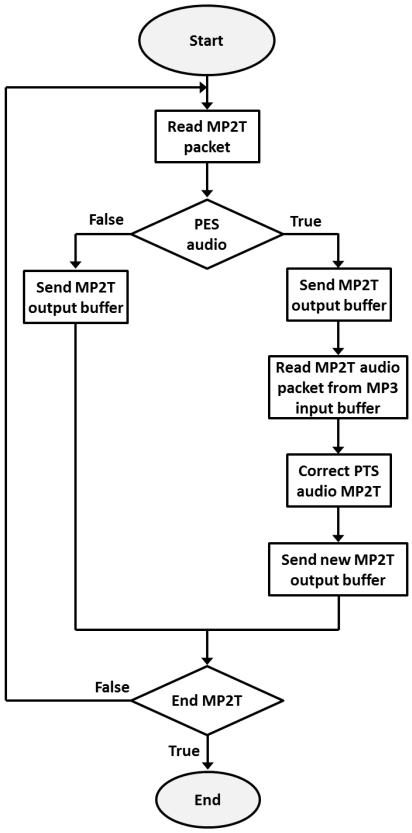


Figure 4.21: MediaSync work-flow for audio substitution replacing original audio with the new audio stream

frame size level. The same level of clock skew has been applied in order to be able to compare this technique with the previous one.

Table 4.8 shows the clock correction levels. If clock skew (ppm) is 125>clock skew>62.5 correction is applied every 414.0s. If clock skew is 187.5>clock skew>125 is applied every 207.0s, and finally, if clock skew>187.5 correction is applied every 138s. A more granular approach could be applied but this was considered unnecessary for a proof-of-concept and in any event, the above logic is similar to the first approach, which facilitated a subjective comparison of approaches.

4.12 Video and Audio Multiplexing (into a single MP2T Stream) and Demultiplexing

As outlined two approaches have been tested in the prototype, audio substitution and audio addition. For ease implementation, both approaches are based on the presumption that the

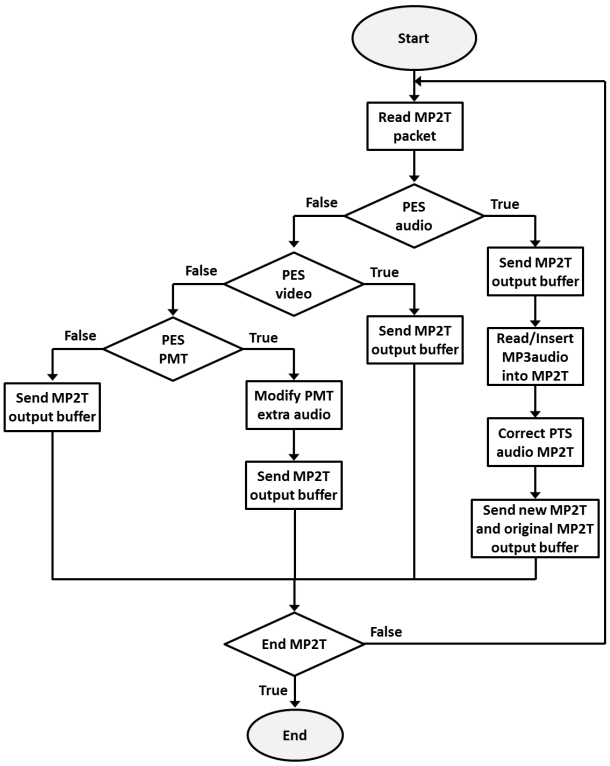


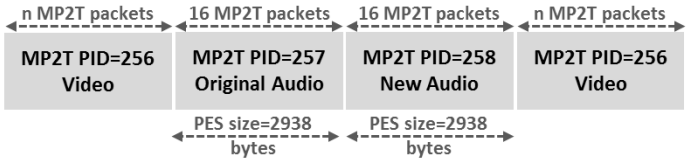
Figure 4.22: MediaSync work-flow for audio addition adding the new audio stream keeping the original one

MP3 audio has the same sampling frequency and audio format as the one within the MP2T stream. Before the application of the MP2T multiplexing used in either of the two techniques, MP3 clock skew detection and correction need to have been applied. Thus, the MP3 audio for addition/substitution has no clock skew relative to the video.

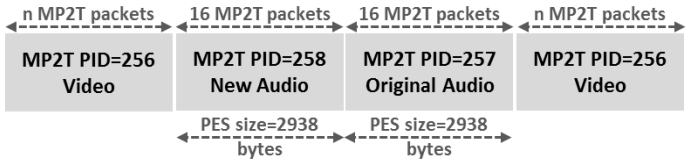
Audio substitution, depicted in Fig. 4.21 replaces the audio stream within the MP2T stream. As outlined previously, the advantages include the fact that the PMT DVB-SI table does not need to be modified whereas the main disadvantage is that the original audio channel is lost.

Audio addition, depicted in Fig. 4.22 adds a new audio stream within the MP2T stream. The advantage is that the original audio channel is kept. The disadvantage is that to add a new audio channel, the PMT DVB-SI table needs to be modified by adding the information for the new audio stream.

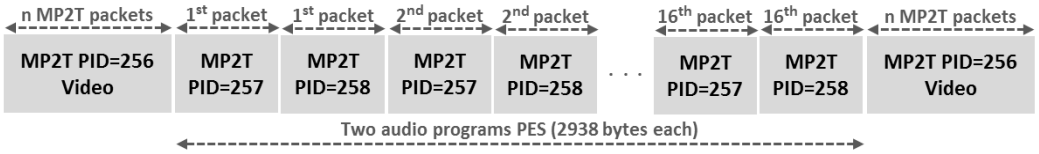
In both Fig. 4.21 and Fig. 4.22, the step to correct the DTS in the PES packets of the audio is not applied in the prototype because the audio characteristics of the MP2T video stream and the MP3 audio are similar. In the case of different characteristics then the correction of the



(a) Insertion of a complete consecutive audio PES within the MP2T.First the original audio PES (PID=257) followed by the new audio PES (PID=258)



(b) Insertion of a complete consecutive audio PES within the MP2T. First the newl audio PES (PID=258) followed by the original audio PES (PID=257)



(c) Insertion of an audio PES interleave with the original audio PES

Figure 4.23: Audio packets distribution in the MP2T stream. Original audio (PID=257) and new added audio (PID=258)

DTS would need to be taken into account using the following equation:

$$x = \frac{newBitrate \cdot DTSoriginal}{128k} \tag{4.13}$$

The MP2T video stream clock skew detection would have been checked prior to the insertion of the MP3 audio. Therefore the video clock skew can be added to that from the MP3 audio and the clock skew result of both of them can be corrected prior to the multiplexing of the new audio within the MP2T video stream. This final step of applying the total clock skews between audio and video and applying the total related clock skew correction has not been applied. As a reminder it is known by MPEG-2 Systems that PLL at receiver corrects the clock frequency in the case of clock skew so it is within the parameters of 27MHz± 810.

Within the audio addition technique, an added consideration needs to be taken into account. This relates to where, within the MP2T stream, the new audio data is to be inserted. Three scenarios have been investigated, the insertion of a complete new audio PES before the original, insertion after the original audio PES, or the insertion of interleaved audio MP2T packets from the original audio and the added audio.

The first scenario is shown in Fig. 4.23a where the new audio PES consisting of 16 MP2T

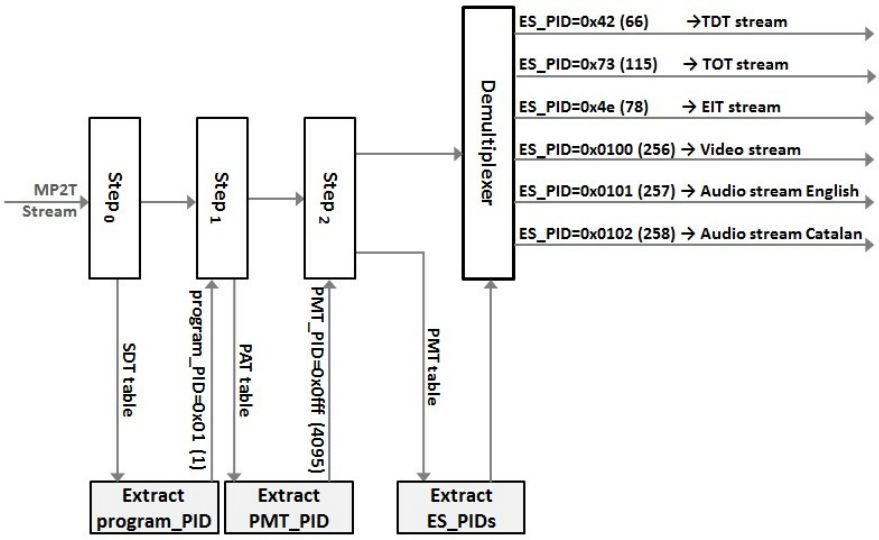


Figure 4.24: High Level demultiplexing structure of DVB-SI and MPEG-2 PSI tables. Following Figure 1.10 in [34]

packets is inserted just after a complete original audio PES. The second scenario is shown in Fig. 4.23b where the new audio PES is inserted just before a complete original audio PES. The final scenario is shown in Fig. 4.23c where the MP2T packets from the two audio PES streams are interleaved.

Fig. 4.24 shows the demultiplexing steps performed on the eventual player when the manipulated MP2T video stream is received. Once the process is finished, different elementary streams are available for decoding.

Firstly, the *program_PID* (MP2T program PID) needs to be extracted from the MP2T video stream. Once the *program_PID* is available, the related PAT table gives the *PMT_PID* which indicates all the elementary streams IDs (*ES_PID*) which relate all the *ES_PID* related to the *program_PID*.

4.13 Summary

This chapter firstly revisited the research questions and outlined a high level architectural solution to address them. It then focused on one particular implementation, and outlined the significant challenges in designing and implementing the proof-of-concept prototype. It presented high level flowchart descriptions of the prototype and then outlined some of the implementation challenges and the range of technologies used. It then outlined in some detail, each of the core prototype components. These include the bootstrapping, the Initial Sync, the MP3 clock detection and correction techniques, the MP2T clock detection and the final MP2T multiplexing that generates the final manipulated MP2T stream with audio addition/substitution. The next

chapter presents a series of results relating to each of these components.

Chapter 5

Prototype Testing

Chapters two to three explained in detail the necessary background information relating to media sync and timelines within different MPEG standards. Chapter 4 outlined the design and implementation details of the proof-of-concept to accomplish the following: the bootstrapping, media stream initial sync, MP3 clock skew detection and correction, MP2T clock skew detection and, finally, the multiplexing of video and audio streams into a single MP2T stream.

This chapter provides details of all testing carried out to evaluate the prototype effectiveness. It is important to note that the scale of testing was limited in that it focused on the technical implementation effectiveness, with some very limited subjective evaluation. Full scale subjective testing would be required to comprehensively evaluate the success of the techniques implemented, and is considered outside the scope of this research, and thus listed as future work. As such, this chapter outlines tests relating to firstly, the Initial Sync of media sync, secondly, the MP3 Clock Skew detection and correction (including results arising from different correction strategies, namely, variable correction over fixed interval, fixed correction over variable interval, and bit correction strategy), thirdly, the MP2T Clock Skew detection and finally, the multiplexing of video and Internet audio channel into a final MP2T stream.

Note that in order to assess the effectiveness of the MP3 and MP2T clock skew detection mechanisms, audio and video files were manipulated to simulate the impact of clock skew on the server-side. Details of such manipulation are also provided in this chapter.

Finally, the chapter concludes by outlining the results of a patent search to assess the extent of patents in this area, and how they relate to the mechanism outlined in this thesis.

5.1 Testing Overview

A Unit Testing approach was deployed to assess the effectiveness of the different stages. These are initial sync, MP3 audio clock skew detection and correction, MP2T clock skew detection and finally the addition of a new audio file within the MP2T video stream.

Firstly, the initial sync was tested to ensure that the initial sport event streamed via IPTV (using RTP) was synchronised with the MP3 audio streamed via Internet Radio. The approach was to sync at the advertised beginning of the game. Whatever time is decided (in the prototype the DVB EIT table is used with the information about the sports event and initial time), both media streams use this to perform the initial sync. The exact time is not important as long as it is agreed.

Secondly, the MP3 audio stream clock skew detection and correction should be tested to ensure that the detection method was accurate enough and the correction technique did not significantly affect audio quality.

Thirdly, the evaluation of the MP2T clock skew detection was tested in order to ensure at the detection mechanism is accurate enough for the accepted clock skew boundaries.

Fourthly, the multiplexing of the new MP3 audio stream within the MP2T video stream be performed in a seamless technique from the user's point of view.

Whilst unit testing is a very useful process, full scale integrated testing is a further necessary step. As outlined, this was not technically feasible, and is further discussed in Section 6.3.

5.2 Testing

5.2.1 Initial Synchronisation

The method outlined in Chapter 4, Section 4.9 was roughly assessed by visually analysing the beginning of the integrated sport event when audio substitution/addition first occurs. As mentioned earlier, more extensive and technical subjective testing would be required to fully evaluate the effectiveness of this mechanism.

In the absence of any skew between the Internet audio stream and IPTV stream, any notable event in the video could also be used to assess the existence of lack of sync. As such, four measurements points were chosen, the beginning of the game, the two goals scored in the first half of the match and the end of the first half. For simplicity, times are shown here to second level granularity, in reality, the synch level operates at a much more precise level, as per synchronisation requirements.

- 00:00:00 → Beginning of the game (20:45:00 wall-clock time)
- 00:26:50 → 1st goal 0-1 scored by Pedro for FC Barcelona (21:11:50 wall-clock time)
- 00:33:04 → 2nd goal 1-1 scored by Rooney for Manchester United (21:18:04 wall-clock time)
- 00:45:02 → End first half of the game (21:30:02 wall-clock time)

From a QoE point of view of the user, no audible lack of sync was detected between the video and the additional Catalan audio stream. The synch mechanism was thus seen to work correctly

ΔPCR(s)	ΔPCR	ΔRTP	PCR/RTP	TR
0.01s	270000	269100	1.003344482	188000.00
0.02s	540000	538200	1.003344482	94000.00
0.03s	810000	807300	1.003344482	62666.67
0.04s	1080000	1076400	1.003344482	47000.00
0.05s	1350000	1345500	1.003344482	37600.00
0.06s	1620000	1614600	1.003344482	31333.33
0.07s	1890000	1883700	1.003344482	26557.14
0.08s	2160000	2152800	1.003344482	23500.00
0.09s	2430000	2421900	1.003344482	20888.89
0.10s	2700000	2691000	1.003344482	18800.00
0.11s	2970000	2960100	1.003344482	17090.91
0.12s	3240000	3229200	1.003344482	15666.66
0.13s	3510000	3498300	1.003344482	14461.54
0.14s	3780000	3767400	1.003344482	13428.57
0.15s	4050000	4036500	1.003344482	12533.33
0.16s	4320000	4305600	1.003344482	11750.00
0.17s	4590000	4574700	1.003344482	11058.82
0.18s	4860000	4843800	1.003344482	10444.44
0.19s	5130000	5112900	1.003344482	9894.737
0.20s	5400000	5392000	1.003344482	9400.00

Table 5.1: Analysis Formula 4 for PCR constant position within MP2T Stream

by identifying the correct start point in the MP3 stream to begin audio addition into the final MP2T stream. Note that the sync levels required for sports commentary are less tight than the requirements of conventional lip-sync shown in Fig. 3.2 in Chapter 3.

5.2.2 Testing MP2T Clock Skew Detection

In Chapter 4, in Section 4.10 the technique used to detect clock skew within an MP2T stream is based on the relationship between PCR and RTP, established in ETSI TS 102 034 [8], as well as the use of RTCP SR packets which establish the relationship between RTP and NTP.

In Table 5.1 the analysis of the formula in ETSI TS 102 034 regarding relationship between RTP and PCR within an MP2T stream is presented. Note that the constant relationship between PCR and RTP values is 1.003344.

Table 5.2 and Fig. 5.1 presents results outlining the extent to which the mechanism correctly detects clock skew via relationship between PCR and NTP values. In order to test the mechanism, varying degrees of skew (from +250 to -250 ppm) were introduced into the MP2T stream at the RTP encapsulation stage within the RTP server thread, and a test was ran for each level. The table illustrates the extent to which this skew level was subsequently detected at the receiving client end. The principal columns of interest from the Table are:

Avg Clock Skew	RTCP N°	RTCP ms	Clock Skew Progress 50	100	150	200	250	300	350	400	450	500	550	Final	Δ CS
-250	562	5522.14	0.8199	0.8136	0.8116	0.8109	0.8105	0.8100	0.8151	0.8069	0.8063	0.8055	0.8049	0.8045	0.0545
-225	562	5530.31	0.8268	0.8289	0.8273	0.8268	0.8265	0.8262	0.8258	0.8255	0.8253	0.8245	0.8232	0.8235	0.0486
-200	566	5503.35	0.8530	0.8469	0.8450	0.8445	0.8440	0.8430	0.8433	0.8430	0.8425	0.8423	0.8451	0.8423	0.0423
-175	565	5491.05	0.8713	0.8444	0.8599	0.8588	0.8568	0.8566	0.8562	0.8230	0.8592	0.8823	0.8100	0.8569	0.0319
-150	560	5500.43	0.8767	0.8708	0.8691	0.8696	0.8693	0.8693	0.8692	0.8696	0.8695	0.8694	0.8696	0.8697	0.0197
-125	557	5525.15	0.8978	0.8916	0.8900	0.8892	0.8884	0.8883	0.8879	0.8884	0.8881	0.8871	0.8880	0.8875	0.0125
-100	560	5496.19	0.9172	0.9115	0.9104	0.9093	0.9087	0.9083	0.9079	0.9086	0.9084	0.9083	0.9085	0.9085	0.0085
-075	564	5517.21	0.9412	0.9341	0.9336	0.9353	0.9355	0.9366	0.9373	0.9381	0.9386	0.9393	0.9394	0.9397	0.0147
-050	564	5524.48	0.9792	0.9736	0.9719	0.9699	0.9676	0.9667	0.9654	0.9650	0.9647	0.9642	0.9638	0.9636	0.0136
-025	558	5512.65	0.9822	0.9759	0.9745	0.9742	0.9740	0.9742	0.9739	0.9743	0.9741	0.9741	0.9742	0.9741	-0.0008
-000	558	5520.42	1.0133	1.0053	1.0028	1.0016	1.0007	1.0004	1.0000	1.0009	1.0007	1.0005	1.0005	1.0004	-0.0004
+025	560	5499.20	1.0377	1.0300	1.0280	1.0273	1.0264	1.0259	1.0257	1.0262	1.0259	1.0256	1.0257	1.0258	0.0008
+050	565	5503.10	1.0812	1.0741	1.0714	1.0700	1.0679	1.0656	1.0652	1.0642	1.0637	1.0631	1.0628	1.0630	0.0130
+075	567	5506.68	1.1013	1.0995	1.0986	1.0975	1.0968	1.0965	1.0961	1.0959	1.0960	1.0955	1.0955	1.0954	0.0204
+100	561	5501.42	1.1248	1.1167	1.1154	1.1150	1.1139	1.1133	1.1129	1.1135	1.1133	1.1134	1.1135	1.1135	0.0135
+125	561	5490.08	1.1550	1.1470	1.1449	1.1438	1.1431	1.1430	1.1430	1.1433	1.1429	1.1427	1.1428	1.1428	0.0178
+150	557	5511.01	1.1842	1.1764	1.1748	1.1734	1.1723	1.1719	1.1713	1.1724	1.1722	1.1721	1.1724	1.1724	0.0224
+175	563	5523.11	1.2374	1.2291	1.2265	1.2258	1.2251	1.2251	1.2251	1.2250	1.2245	1.2239	1.2231	1.2236	0.0486
+200	562	5503.83	1.2775	1.2691	1.2621	1.2592	1.2589	1.2575	1.2562	1.2553	1.2554	1.2549	1.2545	1.2550	0.0550
+225	565	5514.15	1.3193	1.3104	1.3072	1.3068	1.3064	1.3060	1.3053	1.3053	1.3055	1.3051	1.3048	1.3050	0.0800
+250	566	5503.73	1.3634	1.3545	1.3516	1.3515	1.3511	1.3507	1.3501	1.3502	1.3500	1.3493	1.3488	1.3487	0.0987

Table 5.2: Results Positive and Negative MP2T Clock Skew detection applied

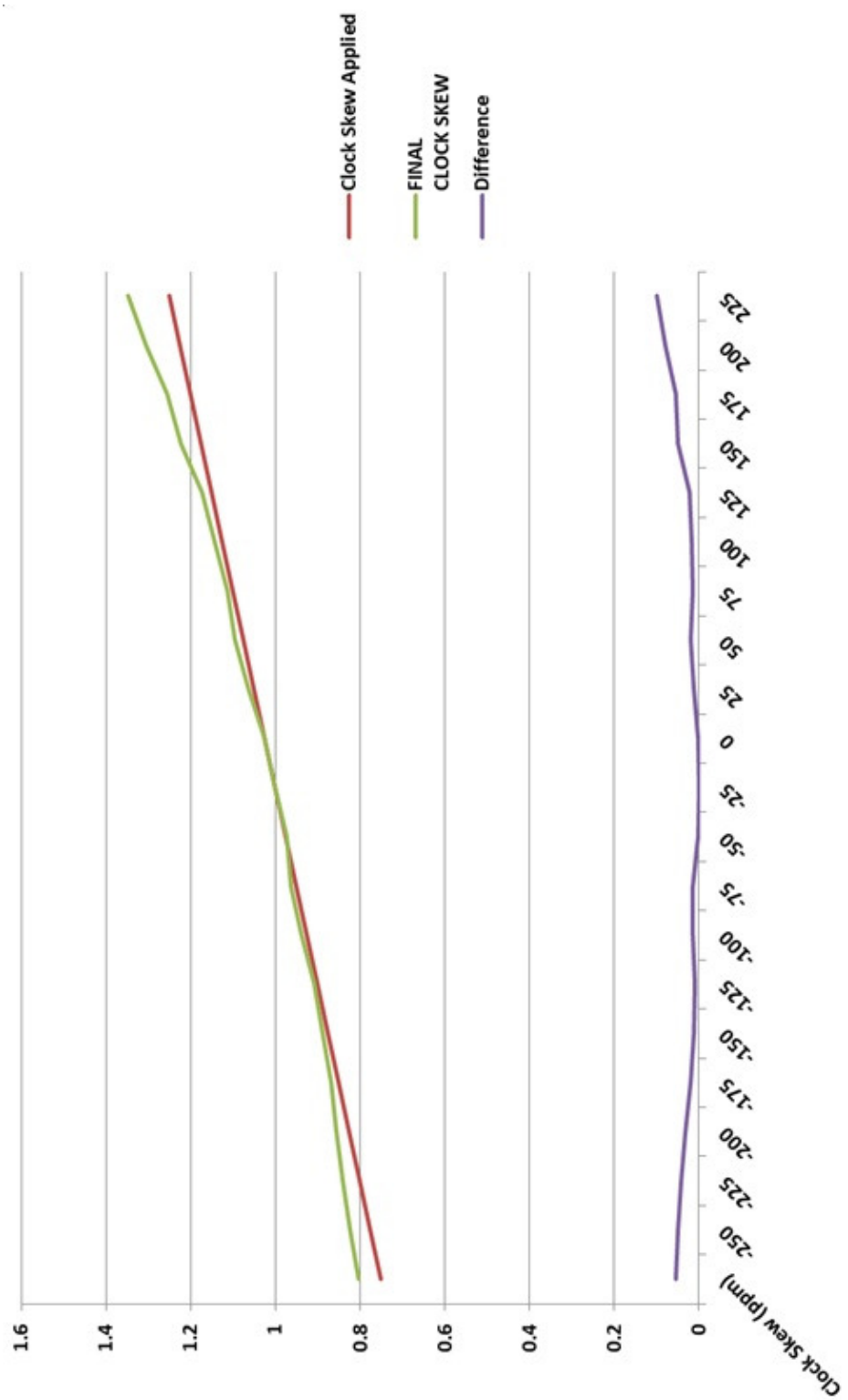


Figure 5.1: Visualisation of result from Table 5.2

- 1^{st} → Skew introduced at server thread
- 2^{nd} and 3^{rd} Column → Number of RTCP packets sent during test and average interval in ms between RTCP packets
- 4^{th} Column → Skew Detection value determined after 50 RTCP packets are received, as expressed as average of consecutive skew values
- 4^{th} to 14^{th} Column → Skew Detection value determined after 100-550 RTCP packets are received
- 15^{th} to 16^{th} Column → Detected Skew after 550 packets expressed as difference from 1 (meaning no clock skew) plus correctness

As expected, whilst there is significant noise in the results though the overall result is very encouraging, with very good correlation between introduced and detected skew levels. Correctness expressed as a % ranges from 75 to 95%. This is especially so as test progresses and timescale over which skew is calculated increases. As outlined in Chapter 4, the full client/server prototype is run on a single laptop as a proof-of-concept. Noise in the dataset is thus expected, due to range of factors including OS non determinism, especially in context of an overloaded device, and thus accuracy improves with test duration. It would be expected that dedicated hardware would eliminate much of this noise.

As previously described this approach needed some manipulation because not all RTP packets convey PCR values and therefore the RTP timestamps of these packets were not used in RTCP packet thread.

5.2.2.1 MP2T clock skew addition to media file at server-side

The process to add at video source clock skew is done by modifying the PCR value with the appropriate clock skew,

$$PCR = PCR \pm \frac{PCR}{29000000} * clockskew \tag{5.1}$$

Using RTCP packets therefore, the relationship between RTP/PCR and NTP can be analysed at the client-side to detect clock skew. In Fig. 4.5 the PCR fields distribution within RTP packets and the distance between two consecutive PCR values are shown.

5.2.3 Testing MP3 Clock Skew Detection and Correction

In Chapter 4 in Section 4.11 the MP3 clock skew detection and correction techniques are described.

Regarding skew detection, a range of approaches was proposed. Detection on a per-frame basis was shown not to be feasible due to the timescales so a per second approach was proposed.

Two separate techniques were outlined - the first using RTP timestamps derived from wall-clock time, in which case detection is based on the difference between elapsed RTP timestamps and bits, and the second whereby RTP timestamps are mapped to the media MP3 rate (similar to VoIP) and using RTCP with the RTP derived from media rate and NTP from system clock. In either case, detection involves comparing bits received (media rate) against elapsed wall-clock time.

Regarding MP3 clock skew correction, two approaches were proposed; variable size correction (1/2/3 bytes) applied every second (fixed time) or fixed size (MP3 frame) correction over a variable frequency (variable time). When the correction is performed every second, a non-rigorous observation suggests that the quality of the audio degrades, by adding a detectable and annoying noise every second. Therefore, this solution was deemed not acceptable.

The second strategy that corrects the clock skew on an MP3 frame basis, modifying the time interval between corrections depending on the clock skew levels.

In order to test the MP3 clock skew detection and correction mechanisms, audio files were manipulated using the Audacity software. This involved simulating skew ultimately resulting in varying file sizes depending on skew. For example, if an MP3 encoder is running fast e.g., +250ppm, then if it runs for 1 TRUE sec, it will generate 1.00025 sec worth of bytes so will be a bigger file. If this file is then played out by a decoder running at the TRUE rate , then it will take 1.00025 sec of true time to play-out, note however a decoder also running fast at 250ppm will play it out in 1sec of true time.

Table 5.3 outlines some key initial results relating to the process of generating test MP3 files to assess the effectiveness of the skew detection process. In summary, it shows the theoretical impact on file size of applying a certain skew level to an MP3 file. It also shows how these theoretical figures were implemented using Audacity, with a small degree of error.

Appendix E lists the tables containing the RTP timestamps values used (Table 19 for negative and Table 19 for positive clock skew).

The first 4 columns columns in Table 5.3 detail the skew level (ppm), the ppm expressed as ms/s, the original file size and its duration. The remaining columns contain the following data:

- MP3 Manipulated File (Theory (Green columns):
  - A → Size in Bytes of MP3 file after applying clock skew
  - B → Absolute value of change in Bytes resulting from clock skew (253789414- Column A)
  - C → Duration in Seconds of MP3 file after applying clock skew if played out by a TRUE MP3 clock, where TRUE implies running with 0 skew.
  - D → Absolute difference in Seconds corrected (Original time - Column C) seconds
- Tempo represents the actual skew level applied using Audacity, which differs slightly from theoretical value.

Clock Skew		Original Values		MP3 Results Theory				Tempo	MP3 Results Audacity				Differences	
ppm	ms/s	Bytes	Sec	A	B	C	D	-	E	F	G	H	I	J
+250	0.250	253789414	15861.83	253852861.4	63447.3535	15865.8038	3.9645	-0.0253	253854196	64782	15865.88	4.0488	-1334.6465	0.0038
+225	0.220	253789414	15861.79	253846516.6	57102.6181	15865.4072	3.5689	-0.0228	253847509	58095	15865.46	3.6309	-992.3818	0.0072
+200	0.200	253789414	15861.83	253840171.9	50757.8828	15865.0107	3.1723	-0.0203	253841657	52243	15865.10	3.2651	-1485.1172	0.0007
+175	0.175	253789414	15861.83	253833827.1	44413.1474	15864.6142	2.7758	-0.018	253834970	45556	15864.68	2.8472	-1142.8525	0.0041
+150	0.150	253789414	15861.83	253827482.4	38068.4121	15864.2176	2.3792	-0.015	253828283	38869	15864.26	2.4293	-800.5879	0.0076
+125	0.125	253789414	15861.83	253821137.7	31723.6767	15863.8211	1.9827	-0.0128	253822849	33435	15863.92	2.0896	-1711.3232	0.0011
+100	0.100	253789414	15861.83	253814792.9	25378.9414	15863.4245	1.5861	-0.0103	253816162	26748	15863.51	1.6717	-1369.0586	0.0045
+75	0.075	253789414	15861.83	253808448.2	19034.2060	15863.0280	1.1896	-0.0078	253809474	20060	15863.09	1.2537	-1025.7939	0.0080
+50	0.050	253789414	15861.83	253802103.5	12689.4707	15862.6314	0.7930	-0.0053	253803623	14209	15862.72	0.8880	-1519.5293	0.0014
+25	0.025	253789414	15861.83	253795758.7	6344.3535	15862.2349	0.3965	-0.0028	253796936	7522	15862.30	0.4701	-1177.2646	0.0049
0	0.00	253789414	15861.83	253789414	0	15861.83	0						0	0
-25	-0.025	253789414	15861.83	253783069.3	6344.3535	15861.4418	0.3965	0.0022	253784815	4599	15861.55	0.2874	-1745.7353	0.0018
-50	-0.050	253789414	15861.83	253776724.5	12689.4707	15861.0452	0.7930	0.0047	253778128	11286	15861.13	0.7053	-1403.4707	0.0052
-75	-0.075	253789414	15861.83	253770379.8	19034.2060	15860.6487	1.1896	0.0072	253771440	17974	15860.71	1.1233	-1060.2060	0.0087
-100	-0.100	253789414	15861.83	253764035.1	25378.9414	15860.2521	1.5861	0.00966	253765589	23825	15860.34	1.4890	-1553.9414	0.0021
-125	-0.125	253789414	15861.83	253757690.3	31723.6767	15859.8556	1.9827	0.0122	253758901	30513	15859.93	1.9070	-1210.6767	0.0056
-150	-0.150	253789414	15861.83	253751345.6	38068.4121	15859.4591	2.3792	0.0147	253752214	37200	15859.51	2.325	-868.4121	0.0090
-175	-0.175	253789414	15861.83	253745000.9	44413.1474	15859.0625	2.7758	0.017	253746781	42633	15859.17	2.6645	-1780.1474	0.0025
-200	-0.200	253789414	15861.83	253738656.1	50757.8828	15858.6660	3.1723	0.0197	253740093	49321	15858.75	3.0825	-1436.8828	0.0060
-225	-0.225	253789414	15861.83	253732311.4	57102.6181	15858.2694	3.5689	0.02224	253732988	56426	15858.31	3.5266	-676.6181	0.0094
-250	-0.250	253789414	15861.83	253725966.6	63447.3535	15857.8729	3.9654	0.0247	253727554	61860	15857.97	3.8662	-1587.3535	0.0029

Table 5.3: Audio files

- MP3 Manipulated File with Audacity (Blue columns):
  - E → Size in bytes of MP3 file after applying clock skew with audacity
  - F → Absolute change in Bytes resulting from clock skew (253789414 - Column E)
  - G → Duration in Seconds of MP3 file after applying clock skew if played out by a TRUE MP3 clock, where TRUE implies running with 0 skew.
  - H → Absolute difference in Seconds corrected (Original time - Column G)
- Comparison between theoretical values and results achieved using Audacity:
  - I → Difference between the number of bytes to be applied in theory (Column A) and the bytes corrected by Audacity (Column E)
  - J → Difference between the number of seconds difference to be applied in theory (Column D) and resulting from Audacity (Column H)

In Table 5.4 and Fig. 5.2, the results indicate the extent to which the prototype was able to detect and correct for the varying degrees of clock skew introduced by audacity from Table 5.3. Table 5.4 is divided into three areas. The green area reproduces the data from audacity as shown above in Table 5.3. The blue area show the values obtained as result of running the prototype with the MP3 files. Finally, the yellow area lists the values generated by the difference between the theory values and the real results obtained.

- MP3 Values from Audacity (Green columns):
  - A → Size in bytes of MP3 file with clock skew applied with audacity
  - B → Duration in seconds of MP3 file with clock skew applied with audacity
  - C → Additional bytes due to application of clock skew with audacity
  - D → Change in seconds due to clock skew
- MP3 Results prototype (Blue columns):
  - E → Actual size in bytes of MP3 audio file with clock skew detection and correction in applied prototype
  - F → Difference in file size bytes between (Column E) and original file (Column A).
  - G → Difference in column F expressed as seconds
  - H & I → Difference in bytes from F expressed in terms of 418 byte and 417 byte frames
- Comparison between manipulated values using audacity and results achieved using the prototype (Yellow columns):

Clock Skew		MP3 Theory values				MP3 Results prototype					Δ Expected and obtained results				
ppm	ms/s	A	B	C	D	E	F	G	H	I	J	K	L	M	%
+250	0.250	253854196	15865.88	63447.3535	4.0488	253806966	47230	2.9518	109	4	1.097	17552	186.0901	63.9008	74.4396
+225	0.225	253847509	15865.46	57102.6181	3.6309	253800279	47230	2.9518	109	4	0.6790	10865	186.0991	38.9008	82.7107
+200	0.200	253841657	15865.10	50757.8828	3.2651	253794427	47230	2.9518	109	4	0.3133	5013	186.0991	13.9008	93.0495
+175	0.175	253834970	15864.68	44413.1474	2.8472	253803623	31347	1.9591	72	3	0.8880	14209	123.5157	51.4842	70.5804
+150	0.150	253828283	15864.26	38068.4121	2.4293	253796936	31347	1.9591	72	3	0.4701	7522	123.5157	26.4842	82.3438
+125	0.125	253822849	15863.92	31723.6767	2.0896	253807383	15466	0.9666	37	0	1.1230	17969	60.9402	64.0597	48.7522
+100	0.100	253816162	15863.51	25378.9414	1.6717	253800696	15466	0.9666	37	0	0.7051	11282	60.9402	39.0597	60.9402
+75	0.075	253809474	15863.09	19034.2060	1.2537	253794008	15466	0.9666	37	0	0.2871	4594	60.9402	14.0597	81.2537
+50	0.050	253803623	15862.72	12689.4707	0.8880	253803623	0	0	0	0	0.8880	14209	0	50	0
+25	0.025	253796936	15862.30	6344.73535	0.4701	253796936	0	0	0	0	0.4701	7522	0	25	0
0	**0**	253789414	15861.83	0	0	253789414	253789414	0	0	0	0	0	0	0	0
-25	-0.025	253784815	15861.55	6344.73535	0.2874	253784815	0	0	0	0	0.2874	4599	0	-25	0
-50	-0.050	253778128	15861.13	12689.4707	0.7053	253778128	0	0	0	0	0.7053	11286	0	-50	0
-75	-0.075	253771440	15860.71	19034.2060	1.1233	253786906	15466	0.9666	37	0	0.1567	2508	-60.9402	-14.0597	81.2537
-100	-0.100	253765589	15860.34	25378.9414	1.4890	253781055	15466	0.9666	37	0	0.5224	8359	-60.9402	-39.0597	60.9402
-125	-0.125	253758901	15859.93	31723.6767	1.9070	253774367	15466	0.9699	37	0	0.9404	15047	-60.9402	-64.0597	48.7522
-150	-0.150	253783564	15859.51	38068.4121	2.325	253752214	31350	1.9593	75	0	0.3656	5850	-123.5276	-26.4723	82.3517
-175	-0.175	253746781	15859.17	44413.1474	2.645	253778131	31350	1.9593	75	0	0.7051	11283	-123.5276	-51.4723	70.5872
-200	-0.200	253740093	15858.75	50757.8828	3.0825	253787327	47234	2.9521	113	0	0.1304	2087	-186.1149	-13.8850	93.0574
-225	-0.225	253732988	15858.31	57102.6181	3.5266	253780222	47234	2.9521	113	0	0.5745	9192	-186.1149	-38.8850	82.7177
-250	-0.250	253727554	15857.97	63447.3535	3.8662	253774788	47234	2.9521	113	0	0.9141	14626	-186.1149	-63.8850	74.4459
										Min:	0.1304	2087	-186.1149	-64.0597	0
										Max:	1.1230	17969	186.0991	64.0597	93.0574
										Avg:	0.6111	9778.7	-0.0035	0.0035	59.4088

Table 5.4: MP3 Clock Skew Detection & Correction - Effectiveness at different Skew rates

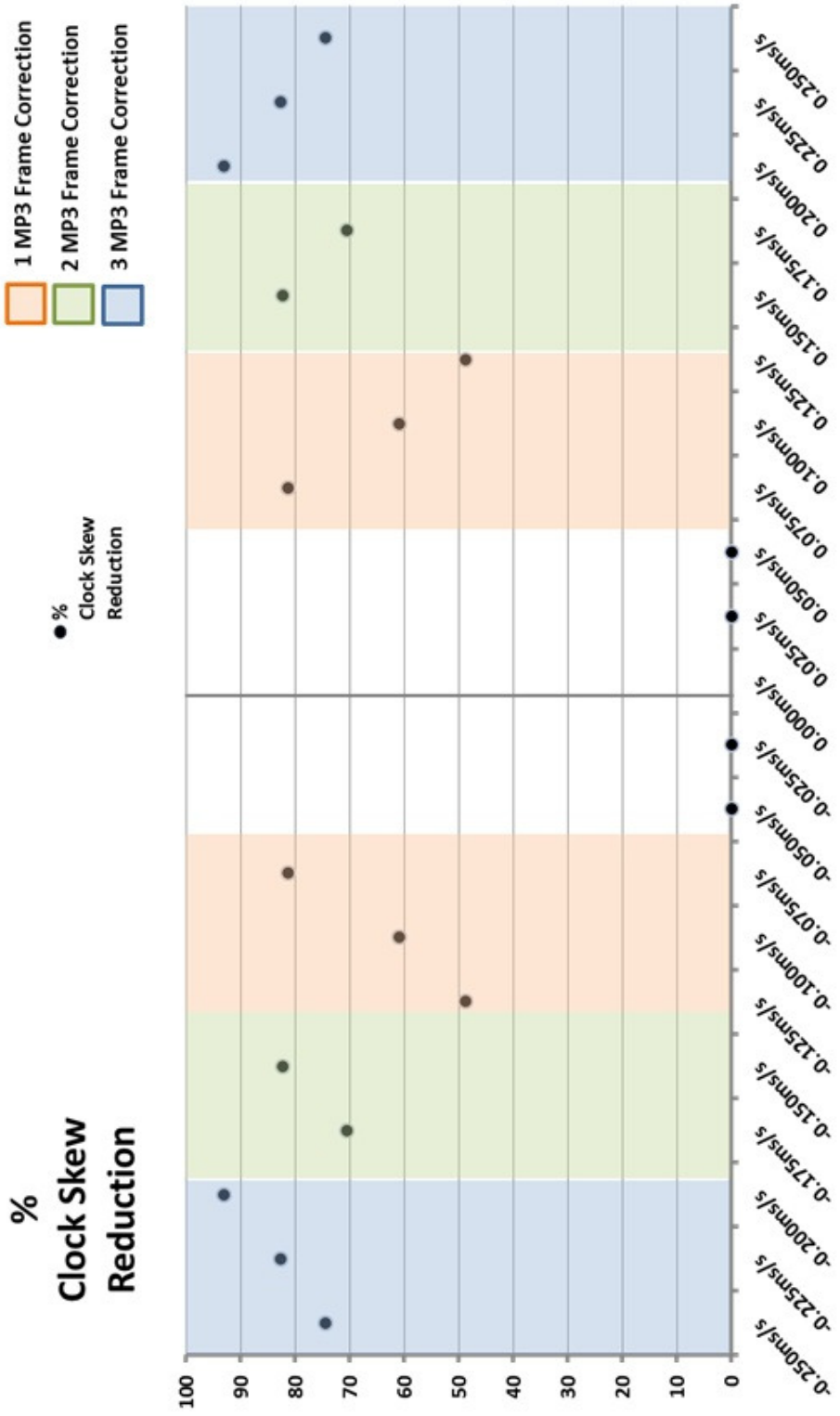


Figure 5.2: Visualisation of the MP3 clock detection and correction results from Table 5.4

- J → Difference between the number of seconds to be applied per audacity and the seconds corrected by prototype
- K → Difference in column J expressed as number of bytes
- L → Actual Clock Skew corrected in prototype
- M → Difference between the Required and Actual applied clock skew
- N & % → Percentage clock skew corrected

As evident from Table 5.4 there is very strong correlation between the desired/required correction and the actual correction applied with correctness values ranging from 48 to greater than 90%. The maximum effectiveness for clock skew detection/correction achieved is 93.0574% when clock skew is ±200 with a difference only of 13 ppm. As a proof-of-concept the results are very promising. The key reasons for the lower effectiveness are likely as follows:

- Stepped Correction: As described in Chapter 4, the correction algorithm applies a stepped adjustment depending on the range of clock skew detected. This was done to ease implementation complexity, and essentially to enable a comparison wit the fixed interval correction approach, and lacks the granularity to achieve more precise results. This is evident in the results from column L above whereby the applied skew correction is similar across a range of differing actual skew values, e.g., for clock skew of 200, 225 and 250 ppm, the applied correction is 186 ppm.
- Prototype Non Determinism: As mentioned previously regarding MP2T skew detection, and detailed at end of the chapter, the entire prototype was implemented on a single device, and thus suffers from non deterministic noise due to Operating System, and Application Software.
- As system timing plays a key role in skew detection, any errors in system clock will manifest in detection errors.

Undoubtedly, the most significant reason for error is the stepped approach in point 1 above, which was simply a design decision to reduce complexity. A more graduated algorithm with finer steps would resolve this issue but in context of thesis scope, the above approach was deemed acceptable.

It is also important to note that the sync thresholds required for live commentary are significantly more relaxed that those for conventional lip-synch as defined and described in Chapter 3.

5.2.4 Multiplexing into a final MP2T stream

As described in Chapter 4, two approaches to integration were proposed, audio addition and audio substitution. Regarding the former, the implementation is much simpler and no testing was needed once the synch issues were addressed. Regarding the latter, described in Section

4.12, a range of integration approaches were proposed in order to embed the additional audio stream within a final MP2T stream. These include placing the full PES of additional audio before the original, after the original, and interleaved on MP2T basis with original.

Regarding the first two approaches, this involved inserting blocks of 16 MP2T audio packets (PID=257) from the added audio before (1^{st}) and after (2^{nd}) the 16 MP2T audio packets from the original audio (PID=258). The structure of this approach is depicted in Fig. 4.23a and Fig. 4.23b (Chapter 4). Based on very small scale non-rigorous subjective testing, and considering the implementation limitations of running full prototype on a single device, the first two approaches added random occasional impairments to the video play-out. The third option, interleaving audio MP2T packets described in Fig. 4.23c (Chapter 4), resulting in no audible noise to audio quality.

5.3 Prototype as proof-of-concept on single device

The prototype has been developed on a single device with NetBeans 7.0.1 with Java platform JDK 1.6. The hardware used (simulating at the same time the two media streamers and the client receiver threads) is a Mac OS X version 6.0.8 with processor 2.3GHz Intel Core i5 with 4G 1333MHz DDR3 memory.

It is important to note that relative to the prototype demands, the equipment suffers from inadequate processing power. For example, in the audio addition scenario, a new PMT table had to be created with the associated recalculated checksum. As all of this was done on the fly, the CRC process took a lot of processing time when done dynamically, resulting in impairment noise in the media file. The problem was solved by storing the CRC checksum when first calculated for the PMT table and then simply using this for subsequent PMT table packets, as the PMT table underwent no further changes.

As a second example, when the MP3 clock skew testing was being executed, it was noticeable that if the prototype logged data to text files, then the total number of bytes of the sent and received audio file did not match. As a result, logging was reduced to minimum and only in the system output window.

Finally, to accomplish/run the prototype correctly the laptop could only be running the prototype and no other applications. Undoubtedly, running the entire PoC introduced significant limitations, and is manifest in noise in the results. However, as a PoC, the results are very promising and auger well for testing in a more professional environment.

5.4 Patent Search

In terms of patentability of the mechanism deployed in the thesis, it is worth reinforcing the point that as Internet Radio does not use RTP protocol (protocol developed in 1996) as a media delivery protocol, the solution per-se is not worthy of patenting. However, it is important to add

that the skew detection mechanism using NTP and RTCP SR is based on a joint NUI,Galway and UCD patent granted in 2009, and was listed as background IP when this PhD research was funded (US patent 7,639,716 - System and method for determining clock skew in a packet-based telephony session).

A search of the patent landscape was carried out to assess the extent to which any other IP has been filed/granted in related areas. This has revealed the following list although the type of media synchronisation performed and/or technology used differs significantly from the thesis implementation.

- WO 1997046027 A1: Preserving synchronization of audio and video presentation.
  Comment: The audio and video are from the same MP2T stream.
- US 20150062429 A1: System and method for video and secondary audio source synchronization.
  Comment: It does not use IP Network as a delivery platform.
- US 5351092 A: Synchronization of digital audio with digital video.
  Comment: it is based on sync at very low level with no consideration of the source of the media used.
- US 7400653 B2: Maintaining synchronization of streaming audio and video using internet protocol.
  Comment: Related to digital cinema network thus not relevant.

As such, none of the above are particularly relevant to the mechanism described in the thesis.

5.5 Summary

This chapter presented a summary of the test results accomplished with the prototype. It included sections dealing with the testing of the Initial Sync process, testing of MP2T Clock Skew detection, MP3 Clock Skew detection and correction and, finally, the final multiplexing into a final MP2T stream.

It is important to re-emphasise that the primary focus of the thesis was to investigate the feasibility of implementing a system that synchronises logically and temporally related media from separate sources on a single end device. As such, this chapter proves the viability of the idea by reporting very positive technical results. However, as stated, the subjective results reporting on the effectiveness of Initial synch, and MP3 skew correction strategies, and on final integration strategies are based on very small scale non rigorous subjective testing, with the additional complications arising from the very limited hardware available. As such, more comprehensive subjective testing on dedicated hardware would be needed for more rigorous results, and this was deemed out of scope. The chapter concluded with a review of the related patent landscape, whereby nothing especially relevant was found.

Chapter 6

Contributions, Limitations, and Future Work

6.1 Introduction

In this thesis, the focus has been on multi-source, multi-platform media synchronisation on a single device. Synchronising multiple media streams over IP Networks from disparate sources opens up a wide range of new potential services. As a sample use case, the PoC focused on live sports events where video and audio streams of the same event are streamed from multiple sources, delivered via IP Networks, and consumed by a single end-device. It aimed to showcase how new interactive, personalised services can be provided to users in media delivery systems by means of media synchronisation.

In meeting the overall thesis objectives, a wide range of challenges and technology choices were discussed. These included the media delivery platforms; TV over IP Network (IPTV) and Internet TV; secondly, multimedia synchronisation; intra and inter as well as multi-source synchronisation, and finally, the technology platform used to receive and deliver the new personalised service to final users.

6.2 Core Contributions

In Section 4.1, the three core research questions to be addressed by the thesis were detailed. These related questions encompass the full life cycle of multimedia from content production, to transport and consumption. More specifically, they were expressed as follows:

1. Given the variety of current and evolving media standards, and the extent to which timestamps are impacted by clock inaccuracies, how can media synchronisation and mapping of timestamps be achieved?

2. Presuming that a mapping between media can be achieved, what impact will different transport protocols and delivery platforms have on the final synchronisation requirement?

3. What are the principal technical feasibility challenges to implementing a system that can deliver multi-source, multi-platform synchronisation on a single device?

Whilst the scope of the PoC prototype was narrow in terms of use case, the overall thesis covers a much broader picture as reflected in the above questions. For example, regarding research Question 1, whilst the PoC was built using MPEG-2 standards, significant research was undertaken into the more recent MPEG-4 standards and how timing is represented. This detailed timing analysis of the current and evolving standards clearly outlined how timing is reflected in the standards.

Regarding Question 2, the thesis examined in detail the various transport protocols and delivery platforms, highlighting their respective strengths and weaknesses. For example, whilst the PoC utilised RTP for Internet Radio delivery to facilitate synchronisation, the thesis also covered evolving standards in the area of HTTP Adaptive Streaming, principally MPEG-DASH, and their approach to timing. As such, the thesis will assist any researcher wishing to see how timing is dealt with within current and emerging standards.

Having dealt with the broader topics, the core practical contribution addressed Question 3 and focused on the design and development of a prototype to showcase the potential for multimedia synchronisation. Despite its significant limitations, discussed shortly, the PoC clearly validates the concept and marks a significant step forward in the area of media synchronisation, relative to other research such as HBB-NEXT and IDES.

The PoC prototype successfully meets the significant challenges of initial synchronisation as well as the skew detection/compensation to ensure that precise media alignment is maintained. The latter involved resolving for relative skew between the RTP/MP3 for audio and RTP/MP2T for video and compensating via manipulation of the audio stream. Whilst margins of error were encountered in skew detection/correction, these were expected and likely due to hardware limitations in the PoC, and were considered acceptable in context of thesis objectives. Similarly, small scale and non rigorous subjective testing was used when assessing various PoC aspects, such as MP3 skew correction, multiplexing of Audio/Video within MP2T.

In terms of broader contribution, the thesis will assist in efforts to promote the significant potential of Time and Timing Synchronisation for Multimedia applications and the challenges in achieving this. The PEL research group at NUI Galway where this thesis was undertaken is strongly aligned with the US-based TAACCS [1] initiative, namely Time Aware Applications, Computers, and Communications Systems. Interest in this concept is growing and in the multimedia field, it has significant potential in Real-time Communications, Massively Multi-player Online Gaming, and pseudo-live streaming.

6.3 Limitations and Future Work

The following section outlines some of the limitations relating to the design and implementation of the PoC. It also identifies a range of areas for possible future work, arising both from these limitations and other issues arising from the thesis scope.

- Moving from PoC to Professional Hardware
  As detailed above, the PoC whilst successful, presented significant technical limitations that undoubtedly impacted on results. It would be very interesting to see how the concepts and techniques would perform in a more professional test-bed environment. Topics of interest might include:
  - Unit versus System Testing: Due to hardware limitations, the PoC was successfully validated using a unit testing approach whereby individual elements/modules within the overall architecture were separately tested. Whilst unit design was done with system integration in mind and thus no significant challenge is foreseen with an integrated system, it would nonetheless be interesting to undertake a complete system test to prove the system.
  - The PoC did not include scalability testing therefore if the idea is taken into a professional scale, this needs to be taken into account. However, even if lots of users demand the service, the synchronisation performed at client-side minimizes the risks. The DVB-IPTV company already are streaming to a large audience therefore the independent clients requiring an Internet Radio stream from Internet should not impact in the system-performance although testing should be performed to corroborate this point.
  - Audio codecs: The PoC utilised MP3 audio that had the same characteristic as the audio within the MP2T video stream therefore no modification of DTS was needed. Further testing would be required to prove the idea using different audio bitrates and/or codecs though this should not present any major issues.
  - Buffering considerations: These have not taken into account in the PoC. In reality, it could be a significant issue due to the time delays in media delivery at the client. Sending the two media streams, video (MP2T stream) and audio (MP3 stream), via RTP and having the servers synchronised via NTP facilitates the calculation of the time difference between servers via the RTCP SR protocol. This will enable the correct buffer size to be determined and facilitate one stream to wait for the other to be received within an allowed time frame to perform the synchronisation.
- Subjective Testing
  Non rigorous and small scale subjective testing was undertaken in assessing certain technology choices in the course of PoC development. A much more rigorous testing was considered out of scope but would make for interesting research.

- The prototype uses RTP as a Media Transport Protocol to simulate the Internet Radio MP3 audio file. As stated, this was done to avail of the RTCP timing support but in reality, such media is streamed via Internet using Adaptive HTTP Protocols therefore the concepts/tools provided by RTP should be adapted to Adaptive HTTP protocol.

- Timing at Source
  It is presumed that the PoC sources have access to, and have implemented, a common time standard such as NTP. Whilst this is a valid presumption, based on the availability of synchronised time due to the wider availability of precision time sources such as GPS, the challenge of ensuring that media content producers deploy common time standards to the required accuracy may not be insignificant. Currently, there is no technical solution to check that media servers are synchronised via NTP to the required level. However, the new RFC 7273 provides some support for such a mechanism. It defines SDP signalling of timestamp reference clock sources and media reference clock sources [69] which is a valid method if the servers are using any of the synchronising methods whereas if it is not signalled, then the receivers assume an asynchronous media clock generated by the sender [69].

- On a related note, the possibilities of using a common UTC timeline between MPEG-DASH and MMT could be investigated, based on the idea that both technologies will be used simultaneously in broadcast and broadband (mainly Internet) delivery platforms.

- Emerging Standards
  In the course of the extensive Literature Review, significant emphasis was placed on emerging standards. As such, future work may involve examining the PoC in light of the more recent MPEG standards timelines; how the time and timing is conveyed and how it is recovered at decoder's side. This will also involve further study of MPEG-DASH and MMT standards. Some ideas include:

  - Regarding MPEG-DASH, issues may include the study of timelines to provide sync with broadcast and broadband media delivery within HbbTV platform. Also, the differences between media containers MP2T and ISO within MPEG-DASH, and performance analysis within an HbbTV platform.
  - MMT has been recently approved and is being used by IPTV and Internet TV. Future research may more deeply analyse timelines within MMT and how it is used in HbbTV environments to sync media streams from different sources using heterogeneous networks delivered via different TV platforms.

6.4 Summary

This chapter concluded the thesis by restarting the core research questions, and reflecting on the extent to which they were addressed. It summarized the core contributions of the thesis addressing also the limitations of the PoC prototype and testing performed. Moreover it describes a range of related future work arising from the thesis.

Appendix A. IPTV Services, Functions and Protocols

A.1 RTP RET Protocol

A.1.1 Retransmission (RTP RET) Architecture

RET refers to the established procedures, unicast and multicast, for retransmission of RTP packets in the event of packet loss. It is defined in ETSI TS 102 034 [8].

The architecture is based is two main elements, an Home Network End Device (HNED) client for both RTP and RTP RET, and the Content on Demand (CoD) or Media Broadcast with Trick Mode [1] (MBwTM) server. The media server could integrate the both RET server and CoD/MBwTM server or two different ones.

RTP RET packets can use the same RTP session with different SSRC identifiers when the RTP and RET servers are the same and use identical transports addresses. The recommendation of using SSRC multiplexing within a single RTP session is by DVB RET. In the event of RTP session multiplexing still SSRC would be different from the RTP stream. On the contrary, RFC 4588 [114] establishes the same SSRC for RTP and RTP RET streams in the case of session multiplexing. Nonetheless different SSRC are used by DVB RET Servers to distinguish at the RTP level, the RTP from the RTP RET streams and monitor the performance of the RET server [8].

There are three different cases in RTP RET, unicast for CoD/MBwTM, unicast for Live Media Broadcast (LMB) and multicast for LMB.

In the first case, unicast solution for CoD/MBwTM depicted in Fig. 1, there are only two nodes involved, a RET client+HNED and a CoD RET+CoD/MBwTM Server. The procedure follows three main steps. First, unicast RTP streaming of CoD/MBwTM media data. Second, when HNED detects packet lost, the HNED/RET Client sends a RTCP Feedback (RTCP FB) message to the CoD RET Server. Finally, the CoD RET Server transmits the RTP RET packet, the retransmitted RTP packet, to the HNED/RET client [8].

[1] Trick mode functions include fast-forward, rewind, pause or slow motion

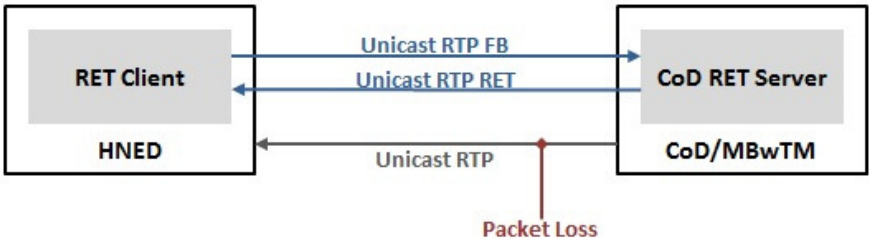


Figure 1: RTP RET Architecture and messaging for CoD/MBwTM services overview. Figure F.1 in [8]

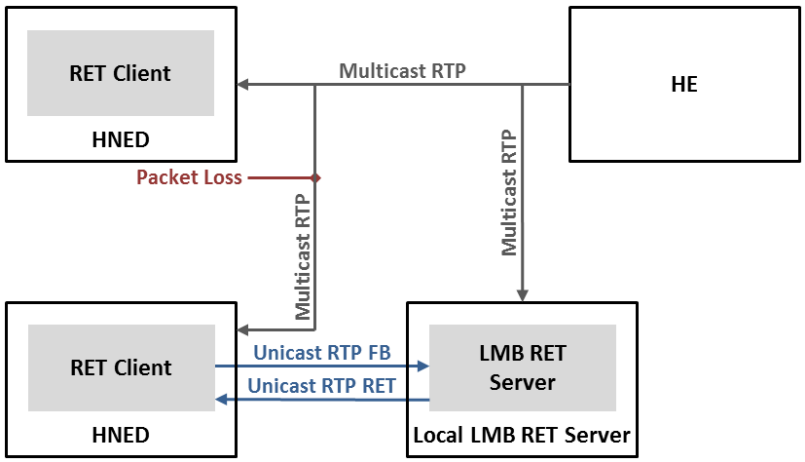


Figure 2: RTP RET Architecture and messaging for LMB services: unicast retransmission. Figure F.2 in [8]

In the second case, unicast solution for LMB, there is an extra node, an independent LMB RET Server involved in the process depicted in Fig. 2, an independent LMB RET server. The procedure follows three main steps. First, multicast RTP streaming of LMB media data. Second, when the HNED Client detects the packet lost, the RET Client sends a RTCP FB to the LMB RET Server which finally, sends via unicast the RTP RET to the HNED/RET Client [8].

In the third case, multicast solution for LMB, as depicted in Fig. 3, the node LMB RET server is also a RET client. The procedure follows three main steps. First, multicast RTP streaming of LMB media data. Second, when the LMB RET Server detects the packet loss the LMB/RET Client sends a RTCP FB to the HE/RET server. Third, the HE/RET server sends the RTP RET packet to the LMB/RET Client which sends the multicast RTP RET to all HNED/RET Clients [8].

A.2 IPTV Services, Functions and Protocols

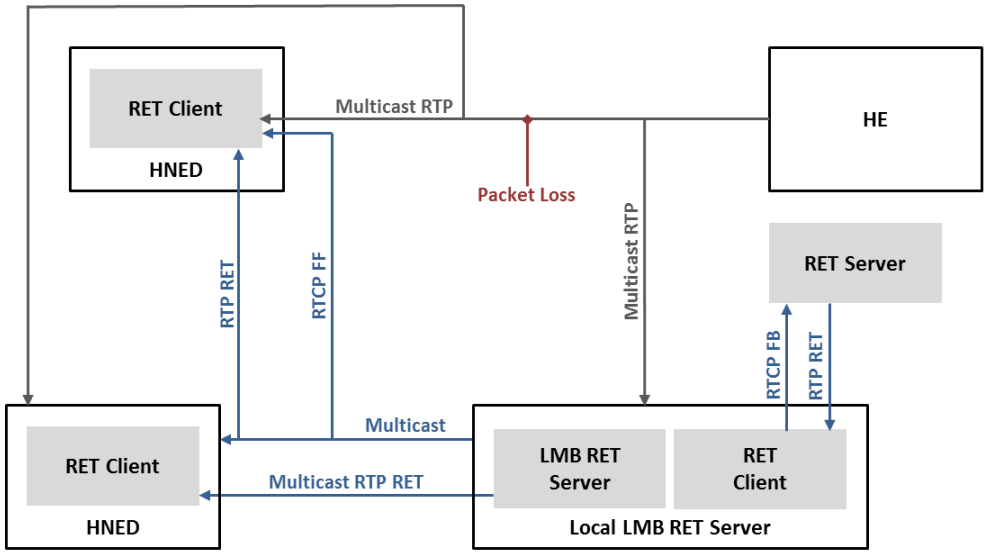


Figure 3: RTP RET Architecture and messaging for LMB services: MC retransmission and MC NACK suppression. Figure F.3 in [8]

Protocol	Function
HTTP	No real-time media delivery
SIP	To stablish, update and end a media session
SDP	To transmit session description information
RTSP	To control media delivery within a media session
IGMP	Multicast Messaging Group to facilitate end-user to join or leave a multicast group
XCAP	A protocol that facilitates the access of configuration information stored using XML
OMA XDM	XCAP and SIP
DVBSTP	Protocol for service access and control functions
RTP	Real-time media delivery
RTP RET	Protocol which facilitates RTP packet retransmission in multicast media delivery systems
SD&S	Service Discovery and Selection
UPnP	server, renderer, controller
DLNA	'*Function is an optional gateway function which serves IPTV content to other DNLA devices in a consumer network*' [6]
DHCP	Protocol to dynamically configure IP address
FLUTE	Protocol for unidirectional file delivery over Internet
RTSP	Protocol for real-time media streaming

Table 1: IPTV Protocols [9]

Service	Description
Scheduled Content Service	Scheduled media delivery streamed at scheduled time for user play-out or recording
CoD or VoD	Media selected from available content for user's play-out or recording
Personal Video Recorder	Scheduled media recording to be stored locally or network-based storage
Time Shift	Service to provide users the option to pause a program and continue the play-out later on
Content Guide	Service to provide user's the program guide with personalize information of the scheduled media programs
Notification Service	Service to provide users information usually notifications and events
Integration with Communication Services	Communications services between users
Web Access	Access to Internet
Information Service	Service to provide all type of information to users not necessarily related to the media delivery
Interactive Applications	Services to provide interactions with user's IPTV Terminal Functions
Parental Control including remote control	Services to provide parents the control over the type of media content accessible for their children
Home Networking	Service to provide DLNA content and on the other hand to provide IPTV services via DLNA
Remote Access	Provide mobile access to Home Network
Support of Hybrid Services	Provide users a personalized content guide
Personalised channel service	Provide users a personalised content guide
Digital Media Purchase	Services to allow users to purchase any type of media
Content sharing	To allow users to share the media under copyrights restrictions

Table 2: IPTV Services based on [6]

Function	Description
Access Networks	Access to fixed or mobile network
Advertising	Provide adverts embedded in multiple services
Content Formats	Shall support standard and high definition media formats
QoS	All services shall be delivered to end users under a minimum QoS
Service Platform Provider	Shall provide authentication, charging and access control functions
Charging	Billing charging functions
Service Usage	Concurrent access to IPTV services
User Interface	Functions to interoperability between end user and IPTV services
User Management	Functions to allow multiple user's accounts
Security	Functions to control user and device access to IPTV services
Services Portability	Functions to access IPTV Services anywhere using multiple ITF devices via multiple network accesses
Services Continuity	Function to provide user the portability of IPTV services over multiple mobile devices
Remote management	Remote performance management, configuration and faults controlling
Content Delivery Networks	Media delivery to end users via multiple media servers
Audience Metrics	Functions to generate and distribute information about the IPTV services' usage
Bookmarks	Functions to marl a point in time within a media stream
Forced Play-out Control	Functions to allow trick mode over media
Remote Control	Functions to provide IPTV services remote control via multiple mobile devices

Table 3: IPTV Functions based on [6]

Appendix B. DVB-SI and MPEG-2 PSI Tables

Field	Bits
service_description_section () {	
table_id	08
section_syntax_indicator	01
reserved_future_use	01
reserved	02
section_length	12
transport_stream_id	16
reserved	02
version_number	05
current_next_indicator	01
section_number	08
last_section_number	08
original_network_id	16
reserved_future_use	08
for (i=0;i<N; i++){	
descriptor()	
service_id	16
reserved_future_use	06
EIT_schedule_flag	01
EIT_present_following_flag	01
running_status	03
free_CA_mode	01
descriptor_loop_length	12
for (i=0;i<N; i++){	
descriptor()	
}	
}	
CRC_32	32
}	

Table 4: SDT (Service Description Section). Table 5 in [40] (SDT Table ID: 0x42)

Field	Bits
event_information_section () {	
table_id	08
section_syntax_indicator	01
reserved_future_use	01
reserved	02
section_length	12
service_id	16
reserved	02
version_number	05
current_next_indicator	01
section_number	08
last_section_number	08
transport_stream_id	16
original_network_id	16
segment_last_section_number	08
last_table_id	08
for (i=0;i<N; i++){	
event_id	16
start_time	40
duration	24
running_status	03
free_CA_mode	01
descriptors_loop_length	12
for (i=0;i<N; i++){	
descriptor()	
}	
}	
CRC_32	32
}	

Table 5: EIT (Event Information Section). Table 7 in [40] (EIT Table ID: 0x4E)

Field	Bits
time_date_section () {	
table_id	08
section_syntax_indicator	01
reserved_future_use	01
reserved	02
section_length	12
UTC_time	40
}	

Table 6: TDT (Time Date Section). Table 8 in [40] (TDT Table ID: 0x70)

Field	Bits
time_offset_section () {	
table_id	08
section_syntax_indicator	01
reserved_future_use	01
reserved	02
section_length	12
UTC_time	40
reserved	04
descriptors_loop_length	12
descriptor_tag	08
descriptor_length	08
country_code	24
country_region_id	06
reserved	01
local_time_offset_polarity	01
local_time_offset	16
time_of_change	40
next_time_offset	16
}	

Table 7: TOT (Time Offset Section). Table 9 in [40] with Local Time Offset Descriptor from Table 67 in [40]. (TOT Table ID: 0x73)

Field	Bits
TS_program_map_section () {	
table_id	08
section_syntax_indicator	01
'0'	01
reserved	02
section_length	12
program_number	16
reserved	02
version_number	05
current_next_indicator	01
section_number	08
last_section_number	08
reserved	03
PCR_PID	13
reserved	04
program_info_length	12
for (i=0;i<N; i++){	
descriptor()	
}	
for (i=0;i<N; i++){	
stream_type	08
reserved	03
elementary_PID	13
reserved	04
ES_info_length	12
for (i=0;i<N; i++){	
descriptor()	
}	
}	
CRC_32	32
}	

Table 8: PMT (TS Program Map Section). Table 2-28 in [30] (PMT Table ID: 0x02)

Field	Bits
program_association_section () {	
table_id	08
section_syntax_indicator	01
'0'	01
reserved	02
section_length	12
transport_stream_id	16
reserved	02
version_number	05
current_next_indicator	01
section_number	08
last_section_number	08
for (i=0;i<N; i++){	
program_number	16
reserved	03
if (program_number=='0") {	
network_PID	13
}	
else {	
program_map_PID	13
}	
}	
CRC_32	32
}	

Table 9: PAT (Program Association Section). Table 2-25 in [30] (PAT Table ID: 0x00)

Appendix C. Clock References and Timestamps in MPEG

C.1 PCR Timestamping

There are two timestamping schemes proposed for the encapsulation of MP2T packets in a ATM, AAL5 scheme [115] [116], PCR-aware and PCR-unaware schemes. These approaches are based on the packetisation distribution of MP2T packets within AAL5 packets. The method establishes the pre-requisite of conveying two MP2T packets within a single AAL5 packet. The PCR-unaware scheme packetises the packets without examining the presence of a PCR field. The PCR-aware technique conveys the two MP2T packets in an AAL5 packet ensuring that any MP2T packet containing a PCR is always encoded in the last packet within the AAL5. The former provides jitter reduction caused by the packetisation process [34] [84].

This effect was firstly named *pattern switch*. On one hand, the PCR-unaware scheme adds packing jitter and, thus, generates an increment of buffer space within the Decoder's time recovery. On the other hand, the PCR-aware technique adds complexity at AAL5 packing stage in order to minimise the packing jitter [117].

In Fig. 4 it can be seen the possible packets structures of two MP2Ts packets within an AAL5 packet following the PCR-unaware scheme. The packets have a constant 384 bytes in total. 188 bytes per each MP2T packet plus an eight byte AAL5 Trailer located at the end of the AAL5 packet.

In Fig. 5 it is shown the possible packets structures of one or two packets within an AAL5 packet following the PCR-aware scheme. The packets can have a 384 bytes like the PCR-unaware scheme or 240 bytes when only one MP2T packet is conveyed within the AAL5 packet, 188 bytes from MP2T packet, 44-byte padding and the 8-byte AAL5 Trailer located at the end of the AAL5 packet. The MP2T Transport over ATM Networks has been extensively studied by Tryfonas [118].

There is an extensive study of both timestamping schemes and the effect on the client's clock recovery [85]. It establishes the classification of MP2T packets, a packet containing a PCR falls in an *odd* boundary if is located the first packet within the AAL5 and if it is located the

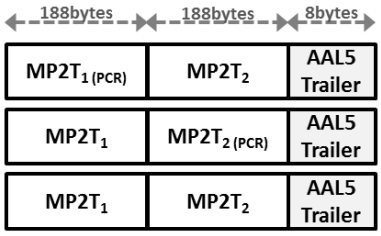


Figure 4: MP2T packetisation scheme PCR-unaware within AAL5 PDUs [117]

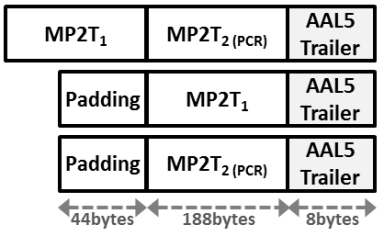


Figure 5: MP2T packetisation scheme PCR-aware within AAL5 PDUs [117]

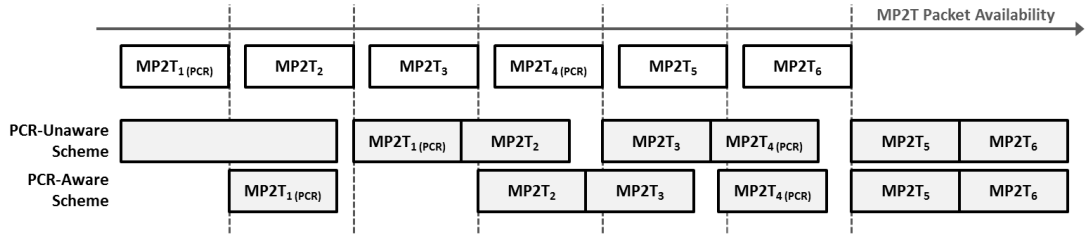


Figure 6: Two PCR packing schemes for AAL5 in ATM Networks. Figure 4.8 in [34]

second packet within the AAL5 then it falls in a *even* boundary. It is highlighted the differences between both schemes in Fig. 6.

Several approaches and their effects have been studied on the clock recovery at decoder. It first analyses the timestamping procedure based in a fixed period timer and then studies the random timestamping scheme [85].

The first approach, based on a fixed period timer, aims to achieve the best quality of the recovered clock based on the timer period and the transport rate. In another words, it aims to find the best pattern switch frequency based on the timer period and the transport rate so PCRs fall into even and odd boundaries in the AAL5 packets at a constant frequency.

The second approach is based on a random timestamping procedure to obtain the lower limits on the rate of change of PCR polarity to achieve the PAL/NTSC specifications at the recovered clock. Three test cases are run. First, to select the deterministic timer period to avoid the phase difference in PCR values, second, fine tuning the deterministic timer period to maximise the pattern switch frequency, and third, the use of random distribution for the timer

period to eliminate the deterministic pattern behaviours.

The results of Tryfonas [85] contradicts the results of Akyildiz [117] by saying that for *'deterministic timer periods, fine-tuning the timer to maximise the frequency of pattern switch results in the best quality of the receiver clock'*.

C.2 Summaries Clock References and Timestamps in MPEG Standards and Media Delivery Techniques

	Standard	Field	Resolution	Frequency	Periodicity	Location
Clock References	MPEG-1	SCR	33-bit	90kHz	0.7s	Pack Header
	MPEG-2 PS	SCR	42-bit	27MHz	0.7s	Pack Header
		ESCR	42-bit	27MHz	0.7s	PES Header
	MPEG-2 TS	PCR	42-bit	27MHz	0.1s	AF Header
		OPCR	42-bit	27MHz	-	AF Header
		ESCR	42-bit	27MHz	0.7s	PES Header
	MPEG-4 SL	OCR	SL.OCRlength (8-bit)	SL.OCRresolution (32-bit)	0.7s [30]	SL Header
	MPEG-4 M4Mux	FCR	FCRlength (8-bit)	FCRresolution (32-bit)	0.7s [30]	M4Mux Packet
Timestamps	MPEG-1	PTS	33-bit	90KHz	-	Packet Header
		DTS	33-bit	90KHz	-	Packet Header
	MPEG-2 PS	PTS	33-bit	90KHz	0.7s	PES Header
		DTS	33-bit	90KHz	-	PES Header
	MPEG-2 TS	PTS	33-bit	90KHz	0.7s	PES Header
		DTS	33-bit	90KHz	-	PES Header
		DTS_next_AU	33-bit	-	-	AF Header
	MPEG-4 SL	CTS	SL.TSlength (8-bit)	SL.TSresolution (32-bit)	-	SL Header
		DTS	SL.TSlength (8-bit)	SL.TSresolution (32-bit)	-	SL Header

Table 10: Clock References and timestamps main differences in MPEG standards (MPEG-1, MPEG-2 and MPEG-4)

	Field	Format	Description
MPD	*availability Start Time*	*xs:dateTime*	For Dynamic type codes the earliest availability of all segments. For Static conveys the segment availability start time. If not present segments availability is equal to the MPD availability
	availability End Time	*xs:dateTime*	Latest availability for all segments. The value it not set when tag is missing
	media Presentation Duration	*xs:duration*	Total duration of the MPD file. Value is not known when not present but it is mandatory when the *minimumUpdatePeriod* field is found
	minimum Update Period	*xs:duration*	The minimum time MPD file to be modified. MPD is not modified when tag is missing and for type Static this field shall not be included
	min Buffer Time	*xs:duration*	Common duration of representation data rate
	time Shift Buffer Depth	*xs:duration*	Time Shifting Buffer guaranteed. For type Dynamic when tag is not included the value is infinite. Value is not defined for Static type
	suggested Presentation Delay	*xs:duration*	For type Dynamic indicates the fixed delay Offset for the AUs presentation time. For type Static the value is not required and if present should be disregarded
	max Segment Duration	*xs:duration*	Establishes the segments maximum duration within the MPD
	max Subsegment Duration	*xs:duration*	Establishes the subsegments maximum duration within the MPD
Period	*start*	*xs:duration*	Indicates the Period start time. It establishes the start time of each Period within the MPD and each AU presentation time in the Media Presentation timeline
	duration	*xs:duration*	Indicates the Period time duration
Segment	*timescale*	xs:unsignedInt	Represents the timescale in units per seconds
	presentation Time Offset		Presentation time offset related to the period's start. Default value is zero.
	duration	*xs:duration*	Conveys the Segment time duration
	Segment Timeline	—	Indicates the segments within the Representation earliest presentation time and duration

Table 11: Time Fields in MPD, Period and Segment within the MPD File [59] [71]

Method	Technology	File Download	Protocols	Drawbacks	Benefits
Downloading	Multiple use	Before play-out	HTPP/TCP IP Unicast	Waiting Time Bandwidth waste	No interrupted play-out No buffer needed
Progressive Downloading	Internet TV	During play-out	HTTP/TCP IP Unicast	Relies on browser Plugins for the play-out	Reduced waiting time
Streaming	IPTV	Along with the play-out	RTP/UDP IP multicast IP Unicast	UDP blocked by firewalls	No waiting Time Low latency Real-time delivery
Adaptive Streaming	Internet TV	Download of small chunks or segments of media during play-out	Multiple protocols	Media content pre-processing (chunks) for various quality formats	Reduced waiting time Adapts to the client's media requirements

Table 12: Media Delivery Techniques from [71]

Appendix D. DVB-SI and MPEG-2 PSI tables in used prototype

Field	Bits		Bytes	Comments
table_id	08	00000010	02	Program_map_section
section_syntax_indicator	01	1	B0 17	
'0'	01	0		
reserved	02	11		
section_length	12	000 00011100		28 bytes
program_number	16	00000000 00000001	00 01	Program Number=1
reserved	02	11	C1	
version_number	05	0000		
current_next_indicator	01	1		
section_number	08	00000000	00	
last_section_number	08	00000000	00	
reserved	03	111	E1 00	
PCR_PID	13	00001 00000000		PCR PID: 256
reserved	04	1111	F0 00	
program_info_length	12	0000 00000000		0 bytes
stream_type	08	00000010	02	13818-2 Video
reserved	03	111	E2 00	
elementary_PID	13	00010 00000000		PID=256
reserved	04	1111	F0 00	
ES_info_length	12	0000 00000000		0 bytes
stream_type	08	00000011	03	11172-3 Audio
reserved	03	111	E1 01	
elementary_PID	13	00001 00000001		PID=257 original audio
reserved	04	1111	F0 00	
ES_info_length	12	0000 00000000		0 bytes
CRC_32	32		F6 4A 03 55	F6 4A 03 55
stream_type	08	00000011	03	11172-3 Audio
reserved	03	111	E1 02	
elementary_PID	13	00001 00000010		PID=258 added audio
reserved	04	1111	F0 00	
ES_info_length	12			
CRC_32	32			

Table 13: PMT fields with three Programs (one video and two audio) in prototype

Field	Bits		Bytes	Comments
table_id	08	01000010	42	Service_description_section
section_syntax_indicator	01	1	B0 25	
reserved_future_use	01	0		
reserved	02	11		
section_length	12	0000 00100101		Section length: 37 bytes
transport_stream_id	16	00000000 00000001	00 01	MP2T ID: 01
reserved	02	11	C1	
version_number	05	00000		
current_next_indicator	01	1		
section_number	08	00000000	00	
last_section_number	08	00000000	00	
original_network_id	16	00000000 00000001	00 01	Network ID: 01
reserved_future_use	08	11111111	FF	
service_id	16	00000000 00000001	00 01	Service ID: 01
reserved_future_use	06	111111	FC	
EIT_schedule_flag	01	0		
EIT_present_following_flag	01	0		
running_status	03	100	80 14	Running Status: 4 running
free_CA_mode	01	0		
descriptor_loop_length	12	0001 00010100		20 bytes
descriptor_tag	08	01001000	48	0x48 service_descriptor
descriptor_length	08	00010010	12	18 bytes
service_type	08	00000001	01	DTV Service MPEG-2 SD
service_provider_name_length	08	00000110	06	6 bytes
service_provider_name	08		46 46 6D 70 65 67	ffmpeg
service_name_length	08	00001001	09	9 bytes
service_name	08		2C ED 12 21	
CRC_32	32			

Table 14: SDT with Service Descriptor in prototype

Field	**Bits**		**Bytes**	**Comments**
table_id	08	00000000	00	Program_association_section
section_syntax_indicator	01	1	B0 0D	
'0'	01	0		
reserved	02	11		
section_length	12	0000		13 bytes
		00001101		
transport_stream_id	16	00000000	00 01	MP2T ID: 01
		00000001		
reserved	02	11	C1	
version_number	05	00000		
current_next_indicator	01	1		
section_number	08	00000000	00	
last_section_number	08	00000000	00	00 01
program_number	16	00000000	00 01	Program Number: 01
		00000001		
reserved	03	111	EF FF	
program_map_PID	13	01111		Program Map PID: 4095
		11111111		
CRC_32	32			

Table 15: PAT fields in prototype

Field	Bits		Bytes	Comments
table_id	08	01001110	4E	Event_information_section
section_syntax_indicator	01	1	E0 47	
reserved_future_use	01	1		
reserved	02	11		
section_length	12	0000		71 bytes
		01000111		
service_id	16	00000000	00 01	Service ID: 1
		00000001		
reserved	02	11	C3	
version_number	05	00001		
current_next_indicator	01	1		
section_number	08	00000001	01	
last_section_number	08	00000001	01	
transport_stream__id	16	00000000	00 01	MP2T ID: 1
		00000001		
original_network__id	16	00000000	00 01	Original Netowork ID: 1
		00000001		
segment_last_section_number	08	00000000	00	Segment Last Section: 0
last_table_id	08	01000010	42	Service Description
event_id	16	00000000	00 01	Event ID: 1
		00000001		
start_time	40		D9 9A	25/05/2011 19:45:00
			19 45	
			00	
duration	24	00000010	02 00	02:00:00 hours
		00000000	00	
		00000000		
running_status	03	100	80 2C	Status running
free_CA_mode	01	0		
descriptor_loop_length	12	0000		44 bytes
		00101100		
descriptor_tag	08	01010100	54	Content_descriptor
descriptor_length	08	00000010	02	2 bytes
content_nibble_level_1	04	0100	43	Sports
content_nibble_level_2	04	0011		Football
user_byte	08	00000000	00	
descriptor_tag	08	01001110	4D	Short_event_descriptor
descriptor_length	08	00100101	26	38 bytes
ISO_639_language_code	24		65 6E	eng
			67	
event_name_length	08	00010011	13	19 bytes
event_name_char				"ChampionsLeague2011"
text_length	08	00001110	0E	14 bytes
text_char				"Barca vs ManU"
CRC_32				

Table 16: EIT fields with Short Event and Content Descriptors in prototype

Field	Bits		Bytes	Comments
table_id	08	1110000	0x70	Time_date_section
section_syntax_indicator	01	1	0xF0	
reserved_future_use	01	1		
reserved	02	11		
section_length	12	0000 00001001		9
UTC_time	40			D9 9A 19 39 25 25/05/2011 19:39:25

Table 17: TDT fields in prototype

Field	Bits		Bytes	Comments
table_id	08	01001011	73	Time_offset_section
section_syntax_indicator	01	1	F0 1A	
reserved_future_use	01	1		
reserved	02	11		
section_length	12	0000 00011010		26 bytes
UTC_time	40		D9 9A 19 39 25	25/05/2011 19:39:25
reserved	04	1111	F0 0F	
descriptor_loop_length	12	0000 00001111		13 bytes
descriptor_tag	08	01011000	58	Time_offset_descriptor
descriptor_length	08	00001101	0D	13 bytes
country_code	24		11 49 52	IRL
country_region_id	06	000000	03	No Time zone extension used
reserved	01	1		
local_time_offset_polarity	01	1		Positive polarity
local_time_offset	16	00000000 00000000	00 00	No offset
time_of_change	40		00 00 00 00 00	
next_time_offset	16	00000000 00000000	00 00	
CRC_32	32			

Table 18: TOT fields with Local Time Offset Descriptor in prototype

Appendix E. RTP Timestamps used in prototype for MP3 streaming

E.1 RTP Timestamps for MP3 Clock Skew Detection

The formula is calculated based in the bits/time relationship of the bitrate which in this case is 128bps per second (1000ms). Thus, the formula used to detect clock skew is:

$$RTP_{timestamp}(x) - RTP_{timestamp}(x-1) = \frac{bitsReceived \cdot 1000}{128000} \tag{1}$$

From the RTP timestamps point of view their relation with clock skew is detailed in the following equations which indicates a clock skew increment of 0.025ms/s is mapped to an increment in the RTP timestamps of 2.25.

$$\begin{cases} 90000 \rightarrow 1s \\ 90 \rightarrow 1ms \end{cases} \tag{2}$$

$$\begin{cases} 90 \rightarrow 1ms \\ x \rightarrow \Delta\ clockSkew \end{cases} \tag{3}$$

$$x = \frac{90 \cdot 0.025}{1} = 2.25\ \Delta\, clockSkew \tag{4}$$

From the bits received point of view their relation with clock skew is detailed in the following equations which indicates a clock skew increment of 0.025ms/s is mapped to an increment in the bitrate of 3.2bps (0.4bytes).

	300	**275**	**250**	**225**	**200**	**175**	**150**	**125**	**100**	**075**	**050**	**025**	**0**
00	2327	2327	2327	2328	2329	2329	2329	2329	2329	2329	2329	2329	2329
01	2327	2327	2327	2328	2329	2329	2329	2329	2329	2329	2329	2329	2329
02	2327	2327	2328	2328	2328	2329	2329	2329	2329	2329	2329	2329	2329
03	2327	2327	2328	2328	2328	2329	2329	2329	2329	2329	2329	2329	2329
04	2327	2328	2328	2328	2328	2328	2329	2329	2329	2329	2329	2329	2329
05	2327	2328	2328	2328	2328	2328	2329	2329	2329	2329	2329	2329	2329
06	2328	2328	2328	2328	2328	2328	2328	2329	2329	2329	2329	2329	2329
07	2328	2328	2328	2328	2328	2328	2328	2329	2329	2329	2329	2329	2329
08	2328	2328	2328	2328	2328	2328	2328	2328	2329	2329	2329	2329	2329
09	2328	2328	2328	2328	2328	2328	2328	2328	2329	2329	2329	2329	2329
10	2328	2328	2328	2328	2328	2328	2328	2328	2328	2329	2329	2329	2329
11	2328	2328	2328	2328	2328	2328	2328	2328	2328	2329	2329	2329	2329
12	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328	2329	2329	2329
13	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328	2329	2329	2329
14	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328	2329	2329
15	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328	2329	2329
16	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328	2329
17	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328	2329
18	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328
19	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328
20	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328
21	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328
22	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328
23	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328
24	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328
25	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328
26	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328
27	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328
28	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328
29	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328
30	2328	2328	2328	2328	2382	2328	2328	2328	2328	2328	2328	2328	2328
31	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328
32	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328
33	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328
34	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328
35	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328
36	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328
37	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328
38	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328
	90783	**90785**	**90788**	**90790**	**90792**	**90794**	**90797**	**90799**	**90801**	**90803**	**90806**	**90808**	**90810**
	90783	90785.25	90787.5	90789.75	90792	90794.25	90796.5	90798.75	90801	90803.25	90805.5	90807.75	90810

Table 19: RTP Timestamps used in prototype. Negative clock skew

	0	**025**	**050**	**075**	**100**	**125**	**150**	**175**	**200**	**225**	**250**	**275**	**300**
00	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328	2329	2329
01	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328	2329	2329	2329
02	2328	2328	2328	2328	2328	2328	2328	2328	2328	2328	2329	2329	2329
03	2328	2328	2328	2328	2328	2328	2328	2328	2328	2329	2329	2329	2329
04	2328	2328	2328	2328	2328	2328	2328	2328	2328	2329	2329	2329	2329
05	2328	2328	2328	2328	2328	2328	2328	2328	2329	2329	2329	2329	2329
06	2328	2328	2328	2328	2328	2328	2328	2328	2329	2329	2329	2329	2329
07	2328	2328	2328	2328	2328	2328	2328	2329	2329	2329	2329	2329	2329
08	2328	2328	2328	2328	2328	2328	2328	2329	2329	2329	2329	2329	2329
09	2328	2328	2328	2328	2328	2328	2329	2329	2329	2329	2329	2329	2329
10	2328	2328	2328	2328	2328	2328	2329	2329	2329	2329	2329	2329	2329
11	2328	2328	2328	2328	2328	2329	2329	2329	2329	2329	2329	2329	2329
12	2328	2328	2328	2328	2328	2329	2329	2329	2329	2329	2329	2329	2329
13	2328	2328	2328	2328	2329	2329	2329	2329	2329	2329	2329	2329	2329
14	2328	2328	2328	2328	2329	2329	2329	2329	2329	2329	2329	2329	2329
15	2328	2328	2328	2329	2329	2329	2329	2329	2329	2329	2329	2329	2329
16	2328	2328	2328	2329	2329	2329	2329	2329	2329	2329	2329	2329	2329
17	2328	2328	2329	2329	2329	2329	2329	2329	2329	2329	2329	2329	2329
18	2328	2328	2329	2329	2329	2329	2329	2329	2329	2329	2329	2329	2329
19	2328	2329	2329	2329	2329	2329	2329	2329	2329	2329	2329	2329	2329
20	2328	2329	2329	2329	2329	2329	2329	2329	2329	2329	2329	2329	2329
21	2329	2329	2329	2329	2329	2329	2329	2329	2329	2329	2329	2329	2329
22	2329	2329	2329	2329	2329	2329	2329	2329	2329	2329	2329	2329	2329
23	2329	2329	2329	2329	2329	2329	2329	2329	2329	2329	2329	2329	2329
24	2329	2329	2329	2329	2329	2329	2329	2329	2329	2329	2329	2329	2329
25	2329	2329	2329	2329	2329	2329	2329	2329	2329	2329	2329	2329	2329
26	2329	2329	2329	2329	2329	2329	2329	2329	2329	2329	2329	2329	2329
27	2329	2329	2329	2329	2329	2329	2329	2329	2329	2329	2329	2329	2329
28	2329	2329	2329	2329	2329	2329	2329	2329	2329	2329	2329	2329	2329
29	2329	2329	2329	2329	2329	2329	2329	2329	2329	2329	2329	2329	2329
30	2329	2329	2329	2329	2329	2329	2329	2329	2329	2329	2329	2329	2329
31	2329	2329	2329	2329	2329	2329	2329	2329	2329	2329	2329	2329	2329
32	2329	2329	2329	2329	2329	2329	2329	2329	2329	2329	2329	2329	2329
33	2329	2329	2329	2329	2329	2329	2329	2329	2329	2329	2329	2329	2329
34	2329	2329	2329	2329	2329	2329	2329	2329	2329	2329	2329	2329	2329
35	2329	2329	2329	2329	2329	2329	2329	2329	2329	2329	2329	2329	2329
36	2329	2329	2329	2329	2329	2329	2329	2329	2329	2329	2329	2329	2330
37	2329	2329	2329	2329	2329	2329	2329	2329	2329	2329	2329	2329	2330
38	2329	2329	2329	2329	2329	2329	2329	2329	2329	2329	2330	2330	2330
	90810	**90812**	**90814**	**90817**	**90819**	**90821**	**90823**	**90826**	**90828**	**90830**	**90832**	**90835**	**90837**
	90810	90812.25	90814.5	90816.75	90819	90821.5	90823.5	90825.75	90828	90830.25	90832.5	90834.75	90837

Table 20: RTP timestamps used in prototype. Positive clock skew

Clock Skew	**Clock Skew**	$\frac{128000 \cdot ClockSkew}{1000}$	**bits/bytes**	**RTPtimestamp clock skew**
250ppm **0.250ms/s**	1000ms → 128000 0.250 → x	$\frac{128000 \cdot 0.250}{1000}$	32.0bps/4.0bytes	90000+22.5=90022.5
225ppm **0.225ms/s**	1000ms → 128000 0.225 → x	$\frac{128000 \cdot 0.225}{1000}$	28.8bps/3.6bytes	90000+20.25=90020.25
200ppm **0.200ms/s**	1000ms → 128000 0.200 → x	$\frac{128000 \cdot 0.200}{1000}$	25.6bps/3.2bytes	90000+18.0=90018.0
175ppm **0.175ms/s**	1000ms → 128000 0.175 → x	$\frac{128000 \cdot 0.175}{1000}$	22.4bps/2.8bytes	90000+15.75=90015.75
150ppm **0.150ms/s**	1000ms → 128000 0.150 → x	$\frac{128000 \cdot 0.150}{1000}$	19.2bps/2.4bytes	90000+13.75=90013.5
125ppm **0.125ms/s**	1000ms → 128000 0.125 → x	$\frac{128000 \cdot 0.125}{1000}$	16.0bps/2.0bytes	90000+11.25=90011.25
100ppm **0.100ms/s**	1000ms → 128000 0.100 → x	$\frac{128000 \cdot 0.100}{1000}$	12.8bps/1.6bytes	90000+09.0=90009.0
075ppm **0.075ms/s**	1000ms → 128000 0.075 → x	$\frac{128000 \cdot 0.075}{1000}$	9.6bps/1.2bytes	90000+06.75=90006.75
050ppm **0.050ms/s**	1000ms → 128000 0.050 → x	$\frac{128000 \cdot 0.050}{1000}$	6.4bps/0.8bytes	90000+04.5=90004.5
025ppm **0.025ms/s**	1000ms → 128000 0.025 → x	$\frac{128000 \cdot 0.025}{1000}$	3.2bps/0.4bytes	90000+02.25=90002.25
000ppm **0.000ms/s**	1000ms →128000 0.000 → x	$\frac{128000 \cdot 0.000}{1000}$	3.2bps/0.4bytes	90000+00.00=90000
-025ppm **-0.025ms/s**	1000ms → 128000 0.025 → x	$\frac{128000 \cdot 0.025}{1000}$	3.2bps/0.4bytes	90000-02.25=89997.75
-050ppm **-0.050ms/s**	1000ms → 128000 0.050 → x	$\frac{128000 \cdot 0.050}{1000}$	6.4bps/0.8bytes	90000-04.5=89995.5
-075ppm **-0.075ms/s**	1000ms → 128000 0.075 → x	$\frac{128000 \cdot 0.075}{1000}$	9.6bps/1.2bytes	90000-06.75=89993.25
-100ppm **-0.100ms/s**	1000ms → 128000 0.100 → x	$\frac{128000 \cdot 0.100}{1000}$	12.8bps/1.6bytes	90000-09.0=89991.0
-125ppm **-0.125ms/s**	1000ms → 128000 0.125 → x	$\frac{128000 \cdot 0.125}{1000}$	16.0bps/2.0bytes	90000-11.25=89988.75
-150ppm **-0.150ms/s**	1000ms → 128000 0.150 → x	$\frac{128000 \cdot 0.150}{1000}$	19.2bps/2.4bytes	90000-13.75=89986.25
-200ppm **-0.200ms/s**	1000ms → 128000 0.200 → x	$\frac{128000 \cdot 0.200}{1000}$	25.6bps/3.2bytes	90000-18.0=89982.0
-225ppm **-0.225ms/s**	1000ms → 128000 0.225 → x	$\frac{128000 \cdot 0.225}{1000}$	28.8bps/3.6bytes	90000-20.25=89979.75
-250ppm **-0.250ms/s**	1000ms → 128000 0.250 → x	$\frac{128000 \cdot 0.250}{1000}$	32.0bps/4.0bytes	90000-22.5=89977.5

Table 21: RTP timestamps. Negative clock skew

$$\begin{cases} 128000 \rightarrow 1000ms \\ \qquad x \rightarrow \Delta\, clockSkew \end{cases} \tag{5}$$

$$x = \frac{128000 \cdot 0.025}{1000} = 3.2bps = 0.4bytes \tag{6}$$

In Table 21 shows the values applied in the prototype which covers the clock skew window frame +0.250ms to -0.250ms per second.

The number of bits received is not related to a fix number of RTP packets or MP3 frames (in prototype every RTP packet conveys one MP3 frame). The MP3 frame size could be 417 or 418 bytes. Therefore, the stream has been analysed and the 128000bps value does not provide an integer number of RTP packets, the closest number of bytes to 128000bps needs 90810 RTP time. The result generates that a multiple of 129152 or 129160 thus the RTP difference is 90810. In Appendix E are included the tables of RTP timestamps increments/decrements in prototype. Clock Skew negative is found in Table 19 and clock skew positive is found in Table 20.

Appendix F. ETSI 102 823

Hybrid Sync solution tables

Syntax	Bits
auxiliary_data_structure () {	
payload_format	04
reserved	03
CRC_flag	01
for (i=0;i<N; i++) {	
payload_byte	08
}	
if (CRC_flag==”1”) {	
CRC_32	32
}	
}	

Table 22: Auxiliary Data Structure. Table 1 in [106]

Syntax	Bits
TVA_id_descriptor () {	
descriptor_tag	08
descriptor_length	08
for (i=0;i<N; i++) {	
TVA_id	16
reserved	05
running_status	03
}	
}	

Table 23: TVA Descriptor. Table 113 in [119]. *descriptor_tag*=0x01

Syntax	Bits
broadcast_timeline_descriptor () {	
descriptor_tag	08
descriptor_length	08
broadcast_timeline_id	08
reserved	01
broadcast_timeline_type	01
continuity_indicator	01
pre_discontinuity_flag	01
next_discontinuity_flag	01
status	03
if (broadcast_type==”0”) {	
reserved	02
tick_format	06
absolute_ticks	32
}	
if (broadcast_type==”1”) {	
direct_broadcast_timeline_id	08
offset_ticks	32
}	
if (prev_discontinuity_flag==”1”) {	
prev_discontinuity_ticks	32
}	
if (next_discontinuity_flag==”1”) {	
next_discontinuity_ticks	32
}	
broadcast_timeline_info_length	08
for (i=0;i<broadcast_timeline_info_length; i++) {	
broadcast_timeline_info_byte	08
}	
}	

Table 24: Broadcast Timeline Descriptor. Table 4 in [106]. *descriptor_tag*=0x02

Syntax	Bits
time_base_mapping_descriptor () {	
descriptor_tag	08
descriptor_length	08
time_base_mapping_id	08
reserved	01
num_time_bases	07
for (i=0;i<numtime_base; i++) {	
time_base_id	08
broadcast_timeline_id	08
}	
}	

Table 25: Time Base Mapping Descriptor. Table 7 in [106]. *descriptor_tag*=0x03

Syntax	**Bits**
content_labelling_descriptor () {	
descriptor_tag	08
descriptor_length	08
metadata_application_format	16
if (metadata_application_format==0xFFFF) {	
metadata_application_format_identifier	32
}	
content_reference_id_record_flag	01
content_time_base_indicator	04
reserved	03
if (content_reference_id_record_flag=='1') {	
content_reference_id_record_length	08
for (i=0;i<content_reference_id_record_length; i++){	
content_reference_id_byte	08
}	
}	
if (content_time_base_indicator==1|2) {	
reserved	07
content_time_base_value	33
reserved	07
metadata_time_base_value	33
}	
if (content_time_base_indicator==2) {	
reserved	01
contentId	07
}	
if (content_time_base_indicator==3|4|5|6|7) {	
time_base_association_data_length	08
for (i=0;i<time_base_association_data_length; i++;){	
reserved	08
}	
}	
for (i=0;i<N; i++){	
private_data_byte	08
}	
}	

Table 26: Content Labelling Descriptor. Table 2.80 in H.222 Amendment 1 [120]

Syntax	Bits
private_data () {	
if (content_time_base_indicator==8) {	
time_base_association_data_length	08
time_base_association_data(){	
reserved	07
time_base_mapping_flag	01
if (time_base_mapping_flag==”1”) {	
time_base_mapping_id	08
} else {	
broadcast_timeline_id	08
}	
}	
}	
if (content_time_base_indicator==9|10|11) {	
time_base_association_data_length	08
for (i=0;i<time_base_association_data_length; i++) {	
time_base_association_data_byte	08
}	
}	
for (i=0;i<N; i++){	
private_data_byte	08
}	
}	

Table 27: Private Data structure. Table 10 in [106]

Syntax	Bits
synchronised_event_descriptor () {	
descriptor_tag	08
descriptor_length	08
synchronised_event_context	08
synchronised_event_id	16
synchronised_event_id_instance	08
reserved	02
tick_format	06
reference_offset_ticks	16
synchronised_event_data_length	08
for (i=0; i<N2; i++) {	
synchronised_event_data_type	08
}	
}	

Table 28: Synchronised Event Descriptor. Table 11 in [106]. *descriptor_tag*=0x05

Syntax	Bits
synchronised_event_cancel_descriptor () {	
descriptor_tag	08
descriptor_length	08
synchronised_event_context	08
synchronised_event_id	16
}	

Table 29: Synchronised Event Cancel Descriptor. Table 12 in [106]. *descriptor_tag*=0x06

Appendix G. Multi bitrate analysis MP2T media files

bps	Audio MP2T Packets							Video MP2T packets					
	PES Size	MP2TxPESPTS$_0$	PTS$_n$	ΔPTS	Packets	Gap audio Packets	PES Size	MP2TxPESPTS$_0$	PTS$_n$	ΔPTS	Packets	Gap audio Packets
64k	2938	16	0	299529404	32915 32914	145404	Min: 0 Max: 2400	2938	16	0	299566800	3600	9254493	Min: 0 Max: 19
80k	2938	16	0	299536457	25862 25862	181754	Min: 0 Max: 2104	2938	16	0	299566800	3600	9254493	Min: 0 Max: 19
96k	2938	16	0	299541159	21159 23511	218105	Min: 0 Max: 1644	2938	16	0	299566800	3600	9254493	Min: 0 Max: 19
112k	2938	16	0	299543510	18808 18809	254456	Min: 0 Max: 1678	2938	16	0	299566800	3600	9254493	Min: 0 Max: 19
128k	2938	16	0	299545861	16457	290807	Min: 0 Max: 1678	2938	16	0	299566800	3600	9254493	Min: 0 Max: 19
160k	2938	16	0	299566800	11755 14106 14107	363494	Min: 0 Max: 1445	2938	16	0	299566800	3600	9254493	Min: 0 Max: 19
192k	2938	16	0	299541159	9404 11755	436196	Min: 0 Max: 1348	2938	16	0	299566800	3600	9254493	Min: 0 Max: 19
224k	2938	16	0	299534106	9404	508895	Min: 0 Max: 1315	2938	16	0	299566800	3600	9254493	Min: 0 Max: 19
256k	2938	16	0	299536457	7053 9403 9404	581594	Min: 0 Max: 1445	2938	16	0	299566800	3600	9254493	Min: 0 Max: 19

Table 30: Analysis MP2T data different MP3 bitrates. Video and audio programs

References

[1] Time-Aware Applications Computers, and Communication Systems, August 2015. URL http://www.taaccs.org. 6, 177

[2] ITU E.800. Definitions of Terms related to Quality of Service. *International Telecommunications Union*, September 2008. 9

[3] P. Le Callet, S. Moller, and A. Perkis. Qualinet White Paper on Definitions of Quality of Experience (2012). *European Network on Quality of Experience in Multimedia Systems and Services COST Action 1003*, March 2013. 9, 10

[4] OIPF Functional Architecture v2.3. Specification, Open IPTV Forum, January 2014. vi, 10, 11, 12

[5] ETSI TS 182 027. v3.5.1. Telecomunications and Internet converged Services and Protocols for Advanced Networking (TISPAN); IPTV Architecture; IPTV functions supported by the IMS subsystem. Technical Specification, European Telecommunications Standards Institute, March 2011. vi, 12, 13

[6] OIPF Services and Functions for Release 2 v1.0. Specification, Open IPTV Forum, October 2008. xii, 12, 183, 184, 185

[7] P. Cesar and K. Chorianopoulos. The Evolution of TV Systems, Content and Users Towards Interactivity. *Foundationds and Trends Human-Computation Interaction*, 2(4): 279–373, January 2009. 12

[8] ETSI TS 102 034. v1.5.1. Digital Video Broadcasting (DVB); Transport of MPEG-2 TS Based DVB Service over IP Based Networks. Technical Specification, European Telecommunications Standards Institute, May 2014. vi, vii, viii, ix, 15, 16, 24, 96, 97, 133, 145, 146, 164, 181, 182, 183

[9] OIPF Release 2. Specification Volume 4 - Protocols v2.1. Specification, Open IPTV Forum, June 2011. xii, 15, 183

[10] OIPF Release 2. Specification Volume 4a - Examples of IPTV Protocol Sequences v2.3. Specification, Open IPTV Forum, January 2014. 15

[11] ISO/IEC 14496-14: Information Technology - Coding of Audio-Visual Objects - Part 14: MP4 File Format. Standard, International Standards Organization (ISO/IEC), 2003. 16

[12] ISO/IEC 14496-12: Information Technology - Coding of Audio-Visual Objects - Part 12: ISO Base Media File Format. Standard, International Standards Organization (ISO/IEC), October 2008. viii, x, 16, 42, 43, 105, 108, 109

[13] Cisco Visual Networking Index: Forecast and Methodology, 2012-2017. White Paper, Cisco, May 2013. 17

[14] Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update 2013-2018. White Paper, Cisco, February 2014. 17

[15] Reciva Internet Radio, August 2015. URL http://www.reciva.com. 17

[16] MS-SSTR. Smooth Streaming Protocol v20150630. Standard, Microsoft Corporation, June 2015. 19

[17] R. Pantos. HTTP Live Streaming. Draft-pantos-http-live-streaming-16. Internet Draft, Internet Engineering Task Force (IETF), April 2015. 19

[18] H. Parmar and M. Thornburgh. Adobe's Real Time Messaging Protocol. Standard, Adobe Systems Incorporated, December 2012. 19

[19] Adobe Systems Incorporated. HTTP Dynamic Streaming, 2015. URL http://www.adobe.com/products/hds-dynamic-streaming.html. 19

[20] HbbTV Specification Version 2.0. Specification, HbbTV Association, August 2015. 19

[21] Computer Electronics Association. CEA-2014-B (ANSI) Web-based Protocol and Framework for Remote User Interface on UPnP Networks and the Internet (Web4CE). Standard, Computer Electronics Association, January 2011. 20

[22] ETSI TS 102 796. v1.2.1. Hybrid Broadcast Broadband TV. Technical Specification, European Telecommunications Standards Institute, November 2012. vi, viii, 20, 21, 22, 23, 131

[23] Ericsson. Press Releases, Corporate. Ericsson to enable global video platform for Telefónica Digital, December 2012. URL http://www.ericsson.com/news/1663941. 20

[24] ETSI TS 102 809. v1.12.1. Digital Video Broadcasting (DVB); Signalling and Carriage of Interactive Applications and Services in Hybrid Broadcast/Broadband Environments. Technical Specification, European Telecommunications Standards Institute, July 2013. x, 21, 22, 23

[25] OIPF Release 1. Specification. Media Formats v2.3. Specification, Open IPTV Forum, January 2014. x, 21, 22, 23

[26] Digital Living Network Alliance (DLNA) Home Networked Devide Interoperability Guidelines - Part 2: Media Formats, ed1.0. Technical Specification, International Electrotechnical Commision, August 2007. 22

[27] H. Shulzrinne, A. Rao, and R. Lanphier. RFC 2326, Real Time Streaming Protocol (RTSP). Standards Track, Internet Engineering Task Force (IETF), April 1998. vi, 24, 25, 26

[28] Data Elements and Interchange Formats – Information Interchange – Representation of Dates and Times. Standard, International Standards Organization (ISO/IEC), 2004. 25

[29] M. Handley, V. Jacobson, and C. Perkins. RFC 4566, SDP: Session Description Protocol. Standards Track, Internet Engineering Task Force (IETF), July 2006. 26

[30] ISO/IEC 13818-1. Information Technology - Generic Coding of Moving Pictures and Associated Audio: Systems. Standard, International Standards Organization (ISO/IEC), December 2000. vi, vii, x, xi, xii, 28, 29, 30, 39, 40, 41, 48, 50, 52, 86, 87, 88, 89, 90, 92, 95, 96, 97, 133, 190, 191, 195

[31] C. Herpel and A. Eleftheriadis. MPEG-4 Systems: Elementary Stream Management. *Signal Processing: Image Communication*, 15(4-5):299–320, January 2000. 31

[32] C. Herpel. Elementary Stream Management in MPEG-4. *IEEE Transactions on Circuits and Systems for Video Technology*, 9(2):315–324, March 1999. 31

[33] ISO/IEC 14496-1. Information Technology. Generic Coding of Audio-Visual Objects. Part 1: Systems (2010E). Standard, International Standards Organization (ISO/IEC), June 2010. vi, vii, x, xi, 31, 32, 33, 34, 36, 37, 38, 99, 100, 102, 103, 104, 105

[34] Xuemin Chen. *Transporting Compressed Digital Video*. Kluwer Academic Publishers, 1st edition, 2002. vi, vii, ix, 37, 83, 84, 85, 90, 91, 93, 94, 95, 160, 192, 193

[35] A. Zambelli. IIS Smooth Streaming. Technical Overview. Technical Report, Microsoft Corporation, March 2009. vi, 44

[36] G. Goldberg. RTP/UDP/MPEG-2 TS as a Means of Transmission for IPTV Streams. , Telecomunications union (ITU), Telecommunication Standardization Sector. Focus Group on IPTV. Source Cisco System Inc., USA, July 2006. 48

[37] A. Basso, G. L. Cash, and M. R. Civanlar. Real-Time MPEG-2 Delivery Based on RTP: Implementation Issues. *Signal Processing: Image Communication*, 15(1-2):165–178, September 1999. 48

[38] A. Basso and S. Varakliotis. Transport of MPEG-4 over IP/RTP. *Transactions on Emerging Telecommunications Technologies*, 12(3):247–255, June 2001. 48

[39] A. MacAulay, B. Felts, and Y. Fisher. IP Streaming of MPEG-4: Native RTP versus MPEG-2 Transport Stream. White Paper, Envivio, October 2005. 48

[40] ETSI EN 300 468 v1.14.1. Digital Video Broadcasting (DVB); Specifications for Service Information (SI) in DVB Systems. European Standard, European Telecommunications Standards Institute, January 2014. vi, x, xii, 48, 49, 50, 51, 52, 187, 188, 189

[41] ETSI TR 101 211 v1.11.2. Digital Video Broadcasting (DVB); Guidelines on Implementation and Usage of Service Information (SI). Technical Report, European Telecommunications Standards Institute, May 2012. x, 52

[42] ISO/IEC 23008-1: 2014. Information Technology - High Efficiency Coding and Media Delivery in Heterogeneous Environments - Part 1: MPEG Media Transport (MMT). Standard, International Standards Organization (ISO/IEC), June 2014. 52

[43] L. Youngkwon, P. Kyungmo, L. Jin Young, S. Aoki, and G. Fernando. MMT: An Emerging MPEG Standard for Multimedia Delivery over the Internet. *IEEE Multimedia*, 20(1):80–85, January-March 2013. vii, 52, 55

[44] Y. Lim. MMT, New Alternative to MPEG-2 TS and RTP. *2013 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, pages 1–5, June 2013. vii, 52, 53, 54

[45] G. Fernando. MMT: The Next-Generation Media Transport Standard. *ZTE Communications*, 10(2):45–48, June 2012. vii, 52, 54, 113

[46] S. Aoki, K. Otsuki, and H. Hamada. Effective Usage of MMT in Broadcasting Systems. *2013 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, pages 1–6, June 2003. vii, xi, 54, 55, 65, 66

[47] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RFC 3550, RTP: A Transport Protocol for Real-Time Applications. Standards Track 3550, Internet Engineering Task Force (IETF), July 2003. vii, x, 55, 56, 57, 58, 59, 60

[48] D. Hoffman, G. Fernando, V. Goyal, and M. Civanlar. RFC 2250, RTP Payload Format for MPEG1/MPEG2 Video. Standards Track, Internet Engineering Task Force (IETF), January 1998. xi, 60, 61, 62, 63, 64, 133, 138, 149

[49] V. Swaminathan. Are we in the Middle of a Video Streaming Revolution. *ACM Transactions on Multimedia Computing, Communications and Applications (TOMM)*, 9(40):1–6, October 2013. 63

[50] V. Paulsamy and S. Chatterjee. Network Converge and the NAT/Firewall Problems. *Proceedings of the 36th Annual Hawaii International Conference on System Sciences*, page 10, January 2003. vii, 64, 65

[51] H. Khifi, J. Gregoire, and J. Phillips. VoIP and NAT/Firewalls: Issues, Travesal Techniques, and a Real-world Solution. *IEEE Communications Magazine*, 44(7):93–99, July 2006. 64, 65

[52] T. Strockhammer. Dynamic Adaptive Streaming over HTTP: Standards and Design Principles. *Proceedings of the 2nd annual ACM Conference on Multimedia Systems (MMSys'11)*, pages 133–144, 2011. 66, 67

[53] L. Beloqui Yuste and H. Melvin. A Protocol Review for IPTV and WebTV Multimedia Delivery Systems. *Journal Communications 2012. Scientific Letters of the University of Zilina*, 2, 2012. xi, 67

[54] C. Mueller, S. Lederer, C. Timmerer, and H. Hellwagner. Dynamic Adaptive Streaming over HTTP/2.0. *2013 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6, July 2013. 68

[55] C. Müller, D. Renzi, S. Lederer, S. Battista, and C. Timmerer. Using Scalable Video Coding for Dynamic Adaptive Streaming over HTTP in Mobile Environments. *Signal Processing Conference (EUSIPCO), 2012 Proceedings of the 20th European*, pages 2208–2212, August 2012. 68

[56] M. Walt, C. Timmerer, and H. Hellwagner. A Test-bed for Quality of Multimedia Experience Evaluation of Sensory Effects. *International Workshop on Quality of Experience, 2009. QoMEx 2009*, pages 145–150, July 2009. 68

[57] C. Müller and C. Timmerer. A VLC Media Player Plugin enabling Dynamic Adaptive Streaming over HTTP. *Proceedings of the 19th ACM International Conference on Multimedia MM'11*, pages 723–726, 2011. 68

[58] I. Sodagar. The MPEG-DASH Standard for Multimedia Streaming over Internet. *IEEE Multimedia*, 18(4):62–67, December 2011. 68

[59] ISO/IEC 23009-1:2012. Information Technology. Dynamic Adaptive Streaming over HTTP (DASH). Part 1: Media Presentation Description and Segment Formats. Standard, International Standards Organization (ISO/IEC), April 2012. vii, xii, 70, 112, 196

[60] J. Ridoux and D. Veitch. Principles of Robust Timing over the Internet. *Queue - Emulators*, 8(4):30–43, April 2010. 73

[61] V. Paxson, G. Almes, J. Mahdavi, and M. Mathis. RFC 2330, Framework for IP Performance Metrics. Informational, Internet Engineering Task Force (IETF), May 1998. 73

[62] Microsoft. Windows Hardware Dev Center Archive. Timers, timer resolution and Development of Efficient Code, June 2010. URL http://download.microsoft.com/download/3/0/2/3027D574-C433-412A-A8B6-5E0A75D5B237/Timer-Resolution.docx. 73

[63] A. S. Tanenbaum and A. Woodhull. *The Minix Book. Operating Systems. Design and Implementation.* Pearson Prentice Hall, 3rd edition, 2006. 73

[64] D. Tsafrir, Y. Etsion, D. G. Feitelson, and S. Kirkpatrick. System Noise, OS Clock Ticks, and Fine-grained Parallel Applications. *In Proceedings of the 19th Annual International Conference on Supercomputing (ICS '05). ACM, New York, NY, USA*, pages 303–312, 2005. 73

[65] P. H. Dana. Global Positioning Systems (GPS). Time Dissemination for Real-Time Applications. *Real-Time Systems. Kluwer Academic Pubishers*, 12(1):9–40, January 1997. 73

[66] D. Mills, J. Martin, J. Burbank, and W. Kasch. RFC 5905, Network Time Protocol Version 4: Protocol and Algorithms Specifications. Standards Track, Internet Engineering Task Force (IETF), June 2010. 73, 74

[67] D. Mills. RFC 4330, Simple Network Time Protocol (SNTP) Version 4 for IPv4, IPv6 and OSI. Informational, Internet Engineering Task Force (IETF), January 2006. 74

[68] K. Correll, N. Barendt, and M. Branicky. Design Considerations for Software only Implementations of the IEEE 1588 Precision Time Protocol. *Conference on IEEE 1588-2002*, pages 1–6, 2005. 74

[69] A. Williams, K. Gross, R. van Brandenburg, and H. Stokking. RFC 7273, RTP Clock Source Signalling. Standards Track, Internet Engineering Task Force (IETF), June 2014. xi, 74, 75, 76, 179

[70] A. J. Mason and R. A. Salmon. Factors Affecting Perception of Audio-Video Synchronization in Television. White Paper WHP174, British Broadcasting Corporation. BBC R&D Publications, January 2009. 76, 81

[71] L. Beloqui Yuste, F. Boronat, M. Montagud, and H. Melvin. Understanding Timelines within MPEG Standards. Unpublished, August 2015. xi, xii, 77, 196, 197

[72] C. Demichelis and P. Chimento. RFC 3393, IP Packet Delay Variation Metric for IP Performance Metrics (IPPM). Standards Track, Internet Engineering Task Force (IETF), November 2002. 77

[73] F. Boronat, J. Lloret, and M. Garcia. Multimedia Group and Inter-stream Sychronization Techniques: A Comparative Study. *Elsevier, Information Systems*, 34(1):108–131, March 2009. xi, 76, 80

[74] J. Le Feuvre and C. Concolato. Hybrid Broadcast Services using MPEG DASH. *Media Synchronization Workshop 2013. Nantes (France)*, October 2013. 78

[75] E. Biersack and W. Geyer. Synchronized Delivery and Play-out of Distributed Stored Multimedia Streams. *Multimedia Systems*, 7(1):70–90, January 1999. xi, 79

[76] R. Steinmetz. Human Perception of Jitter and Media Synchronization. *IEEE Journal on Selected Areas in Communications*, 14(1):61–72, January 1996. 81

[77] ATSC Implementation Subcommittee Finding: Relative Timing of Sound and Vision for Broadcast Operations. Doc. ID-191. Technical Specification, ATSC, June 2003. 81

[78] ETSI TR 103 010 v1.1.1 Speech Processing, Transmission and Quality Aspects (STQ); Synchrnonization in IP Networks - Methods and User Perception. Technical Report, European Telecommunications Standards Institute, March 2007. 81

[79] ITU-R BT.1359. ITU Radio Communciation Sector Relative Timing of Sound and Vision for Broadcasting. Recommendation, International Telecommunications Union, November 1998. vii, 81

[80] M. Montagud, F. Boronat, H. Stokking, and R. van Brandemburg. Inter-destination Multimedia Synchronization: Schemes Use Cases and Standardization. *Multimedia Systems*, 18(6):459–482, November 2012. 81

[81] Rec. ITU-R BT 601-5. Studio Encoding Parameters of Digital Television. Recommendation, ITU International Telecommuncation Union - Radiocommunication Sector, 1995. 82

[82] John Watkinson. *The MPEG Handbook*. Focal Press, New York, 2nd edition, September September 2004. 82

[83] Jerry Whitaker. *DTV Handbook*. Video/Audio Professional. McGraw-Hill, New York, 2001. 82

[84] H. Sun, X. Chen, and T. Chiang. *Digital Video Transcoding for Transmission and Storage*. CRC Press, 1st edition, 2005. vii, xi, 83, 84, 93, 94, 95, 96, 192

[85] C. Tryfonas and A. Varma. Timestamping Schemes for MPEG-2 Systems Layer and their Effect on Receiver Clock Recovery. *IEEE Transactions on Multimedia*, 1(3):251–263, September 1999. vii, 91, 192, 193, 194

[86] ISO/IEC 13818-9. Information Technology - Generic Coding of Moving Pictures and Associated Audio: Part 9: Extension for Real Time interface for systems Decoders. Standard, International Standards Organization (ISO/IEC), December 1996. 96

[87] EBU Recommendation R130 (Unidirectional Transport of Constant Bitrate MPEG-2 TS on IP Network). Recommendation, EBU-UER (European Broadcasting Union), March 2010. 97, 132, 138

[88] XML Schema Part 2: Datatypes Second Edition. URL http://www.w3.org/TR/xmlschema-2/#rf-defn. 110

[89] Multimedia Group of Telecom ParisTech. GPAC Group, August 2015. URL http://download.tsi.telecom-paristech.fr/gpac/DASH_CONFORMANCE/TelecomParisTech/. viii, 111, 112, 113

[90] V. Jung, S. Pham, and S. Kaiser. A Web-based Media Synchronization Framework for MPEG-DASH. *IEEE International Conference on Multimedia and Expo Workshops (ICMEW)*, pages 1–2, July 2014. 112

[91] S. Kwang-deok, J. Tae-jun, Y. Jeonglu, K. Chang Ki, and H. Jinwoo. A New Timing Model Design for MPEG Media Transport (MMT). *2012 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, pages 1–5, June 2012. viii, 113, 114

[92] A.C. Begen, T. Akgul, and M. Baugher. Watching Video over the Web, Part 1: Streaming Protocols. *Internet Computing, IEEE*, 15(2):54–63, March-April 2011. 114

[93] A.C. Begen, T. Akgul, and M. Baugher. Watching Video over the Web, Part 2: Applications, Standardization, and Open Issues. *Internet Computing, IEEE*, 15(3):59–63, May-June 2011. 115

[94] B. Li, Z. Wang, J. Liu, and W. Zhu. Two Decades of Internet Video Streaming: A Retrospective View. *ACM Transactions on Multimedia Computing, Communications and Applications (TOMM)*, 9(33):1–20, October 2013. 115

[95] J. Greengrass, J. Evans, and A. C. Begen. Not All the Packets are Equal, Part 1: Streaming Coding and SLA Requirements. *IEEE Internet Computing*, 13(1):70–75, January-February 2009. 115

[96] J. Greengrass, J. Evans, and A. C. Begen. Not All Paquets are Equal: Part 2: The Impact of Network Packet Loss on Video Quality. *IEEE Internet Computing*, 13(2): 74–82, March-April 2009. 116

[97] M. de Castro, D. Carrero, L. Puente, and B. Ruiz. Real-Time subtitles Synchronization in Live Television Programs. *IEEE 6^{th} International Symposium on Broadband Multimedia Systems and Broadcasting (ISBMSB), 2011*, pages 1–6, June 2011. 116, 117

[98] P. Neumann, J. Qi, and V. Reimers. Seamless Delivery Network Switching in Dynamic Broadcast Terminal Aspects. *2011 IEEE International Symposium on Broadcast Multimedia Systems and Broadcasting (BMSB)*, June 2011. 117

[99] P. Neumann and U. Reimers. Live and Time-shifted Content Delivery for Dynamic Broadcast: Terminal Aspects. *IEEE Transactions on Consumer Electronics*, 58(1):53–59, February 2012. 117

[100] C. Concolato, J. Le Feuvre, and R. Bouqueau. Usages of DASH for Rich Media Services. *Proceedings of the 2nd Annual ACM Conference on Multimedia Systems (MMSys '11). New York, USA*, pages 265–270, 2011. 117

[101] C. Concolato, S. Thomas, R. Bouqueau, and J. Le Feuvre. Synchronized Delivery of Multimedia Content over Uncoordinated Broadcast Broadband Networks. *Proceedings of the 3rd Multimedia Systems Conference (MMSys '12)*, pages 277–232, 2012. 117

[102] R. van Brandenburg, H. Stokking, O. van Deventer, F. Boronat, M. Montagud, and K. Gross. RFC 7272, Inter-destination Media Synchronization (IDMS) using the RTP Control Protocol (RTCP). Standards Track, Internet Engineering Task Force (IETF), June 2014. viii, 118, 119, 120

[103] M. Montagud and F. Boronat. On the use of Adaptive Media Playout for Inter-destination Synchronisation. *IEEE Communications Letters*, 15(8):863–865, August 2011. 119

[104] B. Rainer and C. Timmerer. A Quality of Experience Model for Adaptive Media Playout. *6th International Workshop on Quality of Multimedia Experience (QoMEX)*, pages 177–182, September 2014. 121

[105] B. Rainer and C. Timmerer. Adaptive Media Playout for Inter-destination Media Synchronization. *5th International Workshop on Quality of Multimedia Experience (QoMEX)*, pages 44–45, July 2013. 121

[106] ETSI TS 102 823 v1.1.1 Digital Video Broadcasting (DVB); Specification for the Carriage of Synchronized Auxiliary Data in DVB Transport Streams. Technical Specification, European Telecommunications Standards Institute, November 2005. viii, xi, xii, xiii, 121, 122, 123, 124, 125, 126, 209, 210, 211, 213, 214

[107] HBB-NEXT, Deliverable D.4.3.1, Evaluation: Intermediate Middleware Software Components for Content Synchronisation. Document, HBBTV-NEXT, May 2013. 121

[108] HBB-NEXT, Deliverable D.4.5.1, Evaluation: Final Middleware Software Components for Content Synchronisation. Document, HBB-NEXT, December 2013. 121

[109] HBB-NEXT, Deliverable D.2.3.2, Report on User Validation Results. Document, HBB-NEXT, March 2013. 121

[110] C. Köhnen, C. Kobel, and N. Hellhund. A DVB/IP Streaming Test-bed for Hybrid Digital Media Content Synchronisaton. *2012 IEEE International Conference on Consumer Electronics Berlin (ICCE-Berlin)*, pages 136–140, September 2012. viii, 121, 122

[111] C. Köhnen, N. Hellhund, J. Renz, and J. Müller. Inter-Device and Inter-Media Synchronisation in HBB-NEXT. *Media Synchronization Workshop 2013. Nantes (France)*, October 2013. viii, 121, 122

[112] R. Finlayson. RFC 3119, A More Loss-tolerant RTP Payload Format for MP3 Audio. Standards Track, Internet Engineering Task Force (IETF), June 2001. 133

[113] HBB-NEXT. Next Generation Hybrid Media, April 2015. URL `http://www.hbb-next.eu`. 138

[114] J. Rey, D. Leon, A. Miyazaki, V. Varsa, and R. Hakenberg. RFC 4588, RTP Retransmission Payload Format. Standards Track, Internet Engineering Task Force (IETF), July 2006. 181

[115] B-ISDN ATM Adaptation Layer Specification: Type 5 AAL. Series I: Integrated Services Digital Network I.363.5, ITU-T Telecommunication Standardization Sector of ITU, August 1996. 192

[116] D. Grossman and J. Heinanen. RFC 2684, Multiprotocol Encapsulation over ATM Adaptation Layer 5. Standards Track, Internet Engineering Task Force (IETF), September 1999. 192

[117] I.F. Akyildiz, S. Hrastr, H. Uzunalioglu, and W. Yen. Comparison and Evaluation of Packing Schemes for MPEG-2 over ATM using AAL5. *1996 IEEE International Conference on Communications, 1996, ICC'96, Conference Record, Converting Technologies for Tomorrow's Applications*, 3:1411–1415, June 1996. ix, 192, 193, 194

[118] C. Tryfonas and A. Varma. MPEG-2 Transport over ATM Networks. 192

[119] ETSI TS 102 323 v1.5.1. Digital Video Broadcasting (DVB); Carriage and signaling of TV-Anytime information in DVB Transport Streams. Technical Specification, European Telecommunications Standards Institute, January 2012. xiii, 210

[120] ITU-T Recommendation H.222.0 (2000) Amendment 1: Carriage of metadata over ITU-T Rec H.222.0 — ISO/IEC 13818-1 Streams. Equivalent to ISO/IEC 13818-1 (2000) Amendment 1. Technical Specification, ITU-T Telecommunication Standardization Sector of ITU, 2000. xiii, 212