Lucas Pereira Endres

Development of an MPEG2 Multiplexer compliant with SBTVD Digital TV Standard

Porto Alegre

2014

Lucas Pereira Endres

Development of an MPEG2 Multiplexer compliant with SBTVD Digital TV Standard

Trabalho de conclusão de curso apresentado para obtenção do diploma de Engenheiro Eletricista da Universidade Federal do Rio Grande do Sul.

Universidade Federal do Rio Grande do Sul

Escola de Engenharia

Departamento de Engenharia Elétrica

Supervisor: Altamiro Amadeu Susin

Co-supervisor: André Borin Soares

Porto Alegre

2014

CIP - Catalogação na Publicação

Pereira Endres, Lucas
Development of an MPEG2 Multiplexer compliant with SBTVD Digital TV Standard / Lucas Pereira Endres. -- 2014.
116 f.

Orientador: Altamiro Amadeu Susin.
Coorientador: André Borin Soares.

Trabalho de conclusão de curso (Graduação) -- Universidade Federal do Rio Grande do Sul, Escola de Engenharia, Curso de Engenharia Elétrica, Porto Alegre, BR-RS, 2014.

1. SBTVD. 2. Multiplexação. 3. MPEG2. 4. FFmpeg. I. Susin, Altamiro Amadeu, orient. II. Borin Soares, André, coorient. III. Título.

Elaborada pelo Sistema de Geração Automática de Ficha Catalográfica da UFRGS com os dados fornecidos pelo(a) autor(a).

Lucas Pereira Endres

Development of an MPEG2 Multiplexer compliant with SBTVD Digital TV Standard

Este Trabalho de Conclusão de Curso foi analisado e julgado adequado para a obtenção do título de Engenheiro em Engenharia Elétrica na Universidade Federal do Rio Grande do Sul e aprovado em sua forma final pelo Orientador e pela Banca Examinadora.

Trabalho aprovado. Porto Alegre, _____ de julho de 2014.

Altamiro Amadeu Susin
Orientador e Coordenador, UFRGS
Professor, Doutor, Instituto Politécnico de Grenoble, França.

Rhadam Igor Elias de Miranda
RBS TV
Engenheiro, Fundação Universidade Regional de Blumenau, Brasil.

Sergio Bampi
UFRGS
Professor, PhD, Universidade de Stanford, EUA.

Porto Alegre
2014

Acknowledgements

Ao Professor Altamiro Susin pela possibilidade de trabalhar com este assunto que é de grande inetersse profissional para mim e pelas discussões amistosas nos momentos de bloqueio ao longo do projeto.

A André Borin Soares pelos inúmeros esclarecimentos de questões teóricas da codificação de vídeo e pelo fundamental apoio na revisão do presente texto.

A Marcelo Negreiros pela disponibilidade na instalação e configuração das estações de trabalho, servidores e todos os equipamentos de teste utilizados no projeto.

A toda a equipe da RBS TV pela compreensão das ausências durante o período de estágio e pelo auxílio na compreensão dos aspectos práticos da televisão digital.

Aos colegas do Laboratório de Processamento de Sinais e Imagens, pela compreensão nos momentos em que monopolizei os recursos do laboratório, e em especial a Eduardo da Costa pelas pesquisas preliminares com o FFmpeg.

Resumo

A televisão é o meio de comunicação mais importante no Brasil. No início da última década, decidiu-se digitalizar o sistema de transmissão de TV brasileira. Uma ação do governo unindo pesquisadores de todo o país realizou um estudo sobre vários temas de TV Digital. Codificação de vídeo e áudio, bem como os subsistemas de interatividade, multiplexação, modulação e transmissão estão entre os aspectos estudados do sistema DTV. Como resultado surgiu o SBTVD, o padrão de TV digital brasileiro, que é baseado no padrão japonês de transmissão ISDB-T, e adota os padrões H.264 e HE-AAC para codificação de vídeo e áudio. Hoje em dia, quase todos os países sul-americanos e alguns africanos adotaram o SBTVD. O principal objetivo deste trabalho é desenvolver uma ferramenta que cria um Transport Stream compatível com a norma brasileira ABNT NBR15603, que é baseada na norma ISO/IEC 13818-1. Para alcançar este objetivo, realizou-se extensa pesquisa para compreender os conceitos fundamentais introduzidos pela ISO/IEC13818-1 e as diferenças entre esta e a ABNT NBR15603. Algumas ferramentas existentes geram um Transport Stream compatível com as normas internacionais, mas não conseguem obedecer às especificidades brasileiras. Assim, propõe-se uma versão modificada do aplicativo FFmpeg que agora inclui as estruturas obrigatórias do SBTVD no Transport Stream.

Palavras-chaves: Multiplexação. MPEG2. Transport Stream. SBTVD.

Abstract

Television is the most important broadcast media in Brazil. At the beginning of the last decade, it was decided to digitize the Brazilian TV broadcast system. A government action joining researchers of all around the country carried out a study on several topics of Digital TV. Video and audio coding, along with interactivity and multiplexing, modulation and transmission subsystems are among the studied aspects of the DTV System. What came out is the SBTVD, the Brazilian DTV standard, which is based on the Japanese ISDB-T transmission standard, and adopts H.264 and HE-AAC as video and audio coding standards. Nowadays, almost all South American and some African countries adopted the SBTVD. The main objective of this work is to develop a tool that creates a Transport Stream compliant to the Brazilian standard ABNT NBR15603 which is based on the ISO/IEC 13818-1 standard. To achieve this objective, considerable research was carried out to understand the fundamental concepts introduced by ISO/IEC13818-1 and the differences between this standard and the ABNT NBR15603. Some existing tools generate streams compliant to the international standards but fail to obey the Brazilian specificities. An updated version of the FFmpeg framework is therefore proposed which now includes the mandatory structures of SBTVD in the Transport Stream.

Key-words: Multiplexing. MPEG2. Transport Stream. SBTVD.

List of Figures

List of Tables

List of abbreviations and acronyms

64QAM	Quadrature Amplitude Modulation with 64 Symbols
AAC	Advanced Audio Coding
AAC-LC	Advanced Audio Coding - Low Complexity
ABNT	Associação Brasileira de Normas Técnicas
AF	Adaptation Field
ARIB	Association of Radio Industries and Businesses
CAT	Conditional Access Table
CBR	Constant Bit Rate
CRC	Cyclic Redundancy Check
DCT	Discrete Cosine Transform
DTS	Decoding Time Stamp
DTV	Digital Television
DVB	Digital Video Broadcasting
EBC	Empresa Brasileira de Comunicação
EIT	Event Information Table
EPG	Electronic Program Guide
ES	Elementary Stream
fps	frames per second
HE-AAC	High-Efficiency Advanced Audio Coding
HDTV	High Definition TV
ISDB-T	Integrated Services Digital Broadcasting - Terrestrial
IEC	International Electrotechnical Commission
ISO	International Standards Organization

ITU	International Telegraph Union
ITU-T	ITU Telecommunication Standardization Sector
LATM	Low Overhead Audio Transport Multiplex
LTE	Long Term Evolution
MPEG	Moving Picture Experts Group
NIT	Network Information Table
PAT	Program Association Table
PCI	Peripheral Component Interconnect
PCR	Program Clock Reference
PMT	Program Map Table
PTS	Presentation Time Stamp
PES	Packetized Elementary Stream
PID	Packet ID
PLL	Phase Locked Loop
PS	Parametric Stereo
PS	Program Stream
PSI	Program Specific Information
QPSK	Quadrature Phase-Shift Keying
SBR	Spectral Band Replication
SBTVD	Sistema Brasileiro de Televisão Digital
SDT	Service Description Table
SDTV	Standard Definition TV
SI	System Information
STC	System Time Clock
TOT	Time Offset Table
TS	Transport Stream

UHF	Ultra High Frequency
VBR	Variable Bit Rate
VCO	Voltage Controlled Oscillator

List of symbols

`0xN`	Is the hexadecimal number 'N'
DIV	Integer division with truncation of the result towards $-\infty$
%	Modulus operator. Defined only for positive numbers.
`N d`	Is the decimal number 'N'.

Contents

Disclaimer

This work was developed based on the FFmpeg(FFMPEG, 2014) framework and is subject to the LGPL v2.1 Licence (FREE SOFTWARE FOUNDATION, 2014). The modified part of FFMpeg code is within the 'avformat' library, which is Copyright (C) 2003 of Fabrice Bellard.

The H.264 encoder used to create the sample video streams in this work is licenced under the GPL v2 Licence and is Copyright (C) 2005 of Mans Rullgard (mans@mansr.com).

The author is willing to help whomever is interested in obtaining further information on licencing conditions.

```
FFmpeg is free software; you can redistribute it and/or modify it under
the terms of the GNU Lesser General Public License as published by the
Free Software Foundation; either version 2.1 of the License,
or(at your option) any later version.

FFmpeg is distributed in the hope that it will be useful, but WITHOUT
ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
See the GNU Lesser General Public License for more details.

You should receive a copy of the GNU Lesser General Public License
along with FFmpeg; if not, write to the Free Software Foundation, Inc.,
51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.
```

The source code for this modification is publicly available in the Internet under Git version control system, and may be accessed via the following link:

`https://github.com/nasall2/TCC_ffmpeg`

1 Introduction

Historically, television is present in the homes of most Brazilian citizens and it is the main source of entertainment and information to the population. Over the past 10 years, new digital media such as computer and cell phones are being adopted by the population of different social classes. A recent report (IBGE, 2011) says that in 2011, 69 % of the population had a mobile phone line. Although significant, the participation of this medium is still far below the television, whose coverage area reaches 100 % of the territory via satellite (EMBRATEL, 2014), and around 98% of the population by land (GLOBO, 2014). Television is therefore the main channel of communication available to the general public in Brazil.

Although it has unparalleled coverage to other technologies, television has been traditionally used for unidirectional communication. It is not possible, using the analog TV transmission system, to hand to the viewer interactivity with the content submitted by the broadcaster. Compared to *Internet*, for example, television is at disadvantage in this aspect. It is possible, however, to provide interactive programming if there is a way to return data to the broadcaster. With interactivity, it can be presented to the user program options, and from the user's choice, the TV displays the chosen content. Thus, one could combine the interactivity of *Internet* to television coverage and thereby promote the social inclusion of remote areas without access to infrastructure of high-speed data links, available in the larger cities . This technology is not feasible with the current analog television infrastructure: it is impossible to transmit via a single analog channel more than one video and two audio streams simultaneously.

As a solution to this difficulty, ISO/IEC developed the digital transmission system described by ISO/IEC13818-1 standard, commercially known by the acronym which names the committee formed to draft it, MPEG2, or also known as the ITU-T recommendation H.222. The specification describes standards of encoding and transmitting video, audio and system data. One key feature for television applications is that it is possible to broadcast multiple audiovisual programs simultaneously, thus allowing the user to select which information he wants to receive among several services sent by one single broadcaster, in the same physical channel.

In Brazil, since 1994, private companies and government funded research and technical tests to compare the performance of three digital television systems which were known to be efficient in their countries of origin at that time: ATSC (ATSC, 2014), developed in the United States; DVB-T (DVB PROJECT, 2014), developed by a consortium of companies for use in European countries; and ISDB , developed in Japan by ARIB(ARIB,

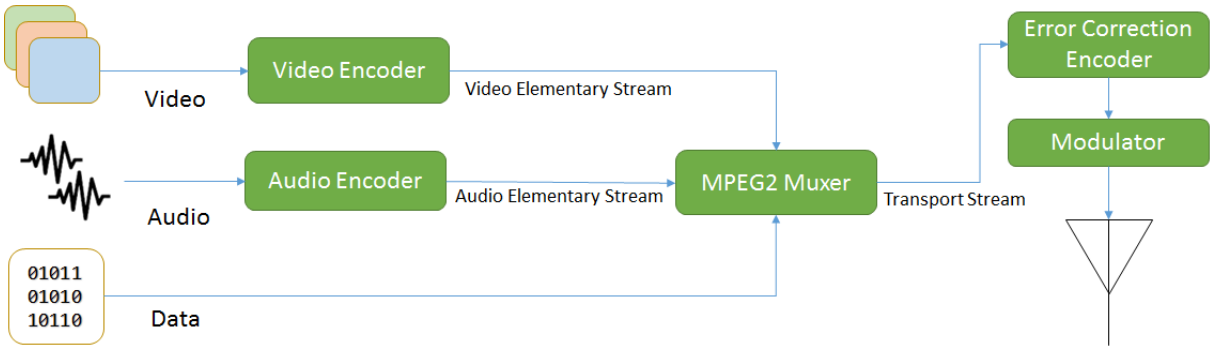
2014). The three systems have similarities and differences regarding the encoding of video and audio: for example, both the DVB and ISDB use the transport infrastructure of MPEG2 standard, but differ in the modulation schemes. After the evaluations, it was determined that the system with the best performance for the Brazilian territory would be based on the Japanese ISDB with some modifications, such as pointed in (BRASIL, 2010).

The differences between the original ISDB and the modified standard used in Brazil mainly relate to video encoding and interactivity platform. In order to promote the development of national industry, the government decided to adopt an open source interactive platform, developed with Ginga, a technology developed mainly by Brazilian researchers(PUC/RJ, 2014). Through this tool the broadcaster can, for example, deliver useful information to the low-income population without access to the *Internet*, as an application sugested by one Brazilian Bank in 2010 (NOW!DIGITAL, 2010). Another application is a project recently developed by the brazilian public comunication company (EBC), called "Brasil 4D" (EBC, 2014). The users can make appointments with doctors or schedule meetings to solve social security issues, or yet check job offers in real time, all this through the TV remote control. However, the similarity of the Brazilian system with the international standard and the fact that interactive interfaces will not be mandatory until 2015 led to the lack of interest of many broadcasters in the development of applications, so that currently very little is invested in the creation of interactive digital television in Brazilian territory.

A digital television system is composed basically by the group of equipments presented in Figure 1. At the beginning of the signal flow, there are the video and audio capturing elements, such as cameras and microphones, and the raw signal may be analog or digital depending on the capture technology used. Once captured, the video and audio are compressed and coded through corresponding encoders. The output of the encoders are standardized bitstreams called Elementary Streams. Once the elementary streams leave the encoders, they enter the multiplexer, where all the streams are packetized and embedded into one single stream, which is the main objective of the MPEG2 systems layer. What follows is one key process for the system robustness: the stream is protected with the use of error correction codes to resist to the noisy, multipath channel between the broadcaster and the receiver. Finally, a digital modulation is applied and the modulated stream is sent to the antenna.

The main difference between analog and digital TV broadcasting systems, at the modulation level, is that in analog TV the multiplexing is in frequency domain, the video signal is sent in a frequency band within the channel bandwidth and the audio in another frequency band. In digital technology, this multiplexing happens in time domain: the multiplexer output has constant bitrate, and for fractions of second only one packet from one of the media sources is sent to the transmitter. In the other side of the channel,

Figure 1 – Block diagram showing digital television basic signal flow.



the receiver is conceived in such a way that after passing the error correction layer, a *demultiplexer* takes the compound stream and splits it back into elementary bitstreams. They are then sent to the decoders and to audiovisual output devices.

It would be certainly disturbing to the viewer to watch delayed video or audio outputs, thus one challenge of this multiplexing scheme is to synchronize the reproduction of all the streams. To ensure this, presentation time stamps are added periodically into the streams so they can be reproduced in sync.

The Presitential Act number 8061(BRASIL, 2010), from july 29, 2013, establishes the chronogram of analog brodcasts shutdown. Until december 31, 2018, all analog transmitters shall be deactivated and the channels shall be vacated. The radio-frequency grants will then be returned to the government, that plans to use the 700MHz band to the 4G mobile telephone service, LTE.

The objective of this project is to develop a simple, yet flexible, tool to handle the Systems layer on SBTVD standard. Basically, by using this tool one is able to produce a SBTVD-compatible binary file containing a Transport Stream that can be used to broadcast recorded video and audio streams into appropriate receivers. Many commercial solutions for this objective are available on the market, so the target here was not to develop a consumer product, but rather something more academic, educational, where most of the configurations can be modified as desired. The main UFRGS internal client is the SBTVD set top box project, which is developing a set-top-box in as a System on Chip architecture in FPGA. In order to run tests of the entire system, streams with different multiplexing setups must be broadcast. When received, the way that the decoder handles them must be analysed.

The project work-flow was the following. The first step was to study the ABNT NBR15603 standard and the references in it to the ISO/IEC 13818 standard and to the ARIB STD-B10 standard. Here the target was to understand how the Systems Layer should work and identify which parts were mandatory and optional. While discussing the

project at the very first weeks, it was decided to evaluate the possibility of develop the tool from scratch. This was before the first step was accomplished, because other people in the lab had already done some research and had concluded that the available freeware tools couldn't handle the SBTVD characteristics. During the author's own research on the Internet, some interesting tools were identified, and instead of developing from scratch, it seemed by then more reasonable to profit of an existent framework.

Once it was decided that an existing tool would be modified, the third step was to add the missing features in order to make it compatible with the SBTVD, and finally the features were tested on several receivers. The practical objective of the project is to provide a tool that creates Transport Streams with three key features: synchronous audiovisual presentation, compliance to the SBTVD standard and multiple services. Eventually, if any mandatory structures are only informational and do not block the system from working, they might be left for further development.

2 Audiovisual codecs used in SBTVD

The codecs chosen to be used in SBTVD are H.264 for video and AAC/LATM for the audio. This overview is useful to understand the synchronization methods adopted in the systems layer, explained further in the text.

2.1 H.264

The H.264 video coding standard is also known as the MPEG-4 Part 10 standard, and it was published as the evolution of the existing video coding standards (ITU-T H.261 / MPEG1, ITU-T H.262 / MPEG2). The main applications for this standard, as described by the publisher itself, are video-conferencing, digital media storage, television broadcasting, Internet streaming and Internet communication. It was developed to provide a solution that has higher compression rates than the former standards, and yet improves subjective quality. To the television broadcast market, it suits perfectly the transmission of high definition video. Technically, the encoder takes the raw video frames and sequentially performs prediction, transform and entropy encoding processes to produce a compressed bitstream. The codec features are organized in profiles and levels, the *high* profile suits the HDTV video transmission characteristics on the Full Segment transmission, while the *baseline* profile fits the needs for mobile, 1-Segment broadcasts. See Appendix A.

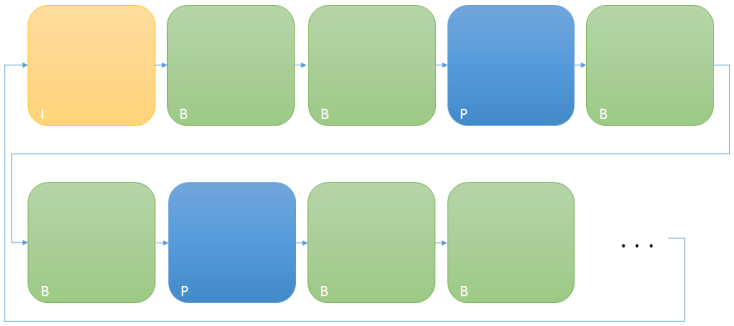
In the first step, prediction, the encoder splits one frame into square or rectangular blocks of pixels, called macroblocks, and predicts the values of pixels in the current macroblock based on information from previously analysed macroblocks either in the current frame or in previously coded frames. Then, instead of transmitting the pixel values, is transmits the difference between the original and the predicted frame (which should have less information if the encoder is efficient) along with pointers to the macroblocks that must be used to recreate that region. These pointers are called motion vectors. There are three different types of prediction schemes, that lead to different kinds of frames:

- an I-frame is constructed by predicting pixel values only from macroblocks within the same frame, this is the intra-frame prediction;
- a P-frame is composed of predictions from the current source frame and also of the previously coded I-frame, this is the inter-frame prediction;
- a B-frame is made by predicting the output frame from the current input, the earlier inputs or yet the later input frames, and is known as the bidirectional inter-frame prediction.

Figure 2 – Visualization of motion compensation encoder working.

Source: (RICHARDSON, 2002).

Figure 3 – Usual group of pictures arrangement.



Source: The Author.

The decrease in amount of information can be noticed in Figure 2, adapted from (RICHARDSON, 2002). The first frame is marked as (a) and coded with intra prediction. The following frame in presentation sequence is (b), which is coded as a P-frame. There is a slight difference between frames that can be seen by the bicycle position. When generic motion compensation coding is applied, the residual frame is the one in (c). Knowing that these frames have differential pixel values [1], it can be noticed that most of the pixels have a value close to zero in (c), which means a residual with very little information. The number of bytes needed to code one I frame is therefore much higher than the necessary to code a P frame, since there is no motion compensation in the first one.

Although there is a substantial decrease in amount of information with inter-frame prediction, the video quality may degenerate if too many P and B frames are used between two I frames, due to quantization. This leads to a compromise of quality and bitrate and the usual frame arrangement is the one shown in Figure 3, that places at most two B frames between one I and one P, or between two P frames, and at most two P frames between consecutive I frames. This arrangement is called a Group of Pictures, or GOP as referenced in MPEG4 standard.

The next encoder step is to transform the residual frame from prediction with an

[1] Differential pixel values means that a black pixel is the negative threshold, a white pixel is the positive threshold and gray is zero.

integer approximation of the Discrete Cosine Transform, or DCT, so that the pixel values become a set of frequency domain coefficients. These coefficients are then quantized, which reduce their precision and make many of them to be equal to zero. Finally, the coefficients are reordered to align most of the zero-valued ones consecutively, and an entropy encoder reduces the redundancy of the sequences of repetitions, hence reducing the amount of data. The output of the H.264 encoder is formed of the quantized coefficients and the motion vectors arranged in a structured bitstream, that constitutes the video Elementary Stream. Further information can be found in (VCODEX, 2014) or (RICHARDSON, 2002).

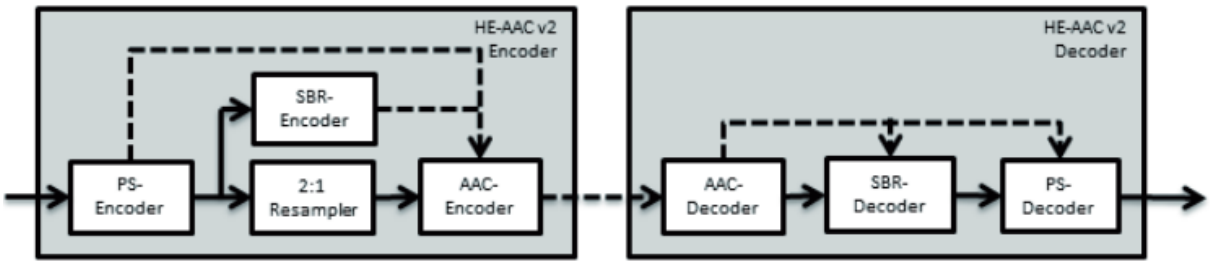
2.2 AAC

The audio codec chosen to incorporate the SBTVD, developed by Fraunhofer IIS along with AT&T, Dolby and Sony, is called Advanced Audio Coding, or AAC. Since the beginning of the development in 1994, the successive evolutions in the codec gave place to profiles that assemble the main features. Three different profiles are the mostly used in digital television broadcast: AAC-LC, HE-AAC, HE-AACv2. According to the main developer(FRAUNHOFER-GESELLSCHAFT, 2013), even the simplest profile, AAC-LC (LC for low complexity) provides *"an audio whose quality even for expert listeners, so-called golden ears, is indistinguishable from the original"*.

The codec supports encoding of ancillary data along with the audio samples, and all with typically 192Kbps for broadcast quality stereo streams in AAC-LC profile. If the HE-AAC profile is chosen, the Spectral Band Replication(SBR) feature is applied and for the same audio quality in the output, a 30% bitrate reduction is observed. In HE-AAC the lower part of the audio spectrum is sampled at half the frequency of the remaining signal using AAC-LC, and the high frequency band is coded with SBR. SBR exploits the relationship of upper and lower sides of the spectrum to produce a parametric recreation of the upper band, thus reducing the total bitrate. HE-AAC achieves typical 48 to 64 kbps for stereo streams.

The chosen profile to integrate the SBTVD is HE-AACv2. Additionally to the already described features, it includes the Parametric Stereo (PS) tool. Rather than encoding two separate channels without any relationship between them, PS creates a monaural downmix of the two stereo samples and a set of parametric coefficients. The mono stream is then encoded with HE-AAC and sent along with the coefficients to recreate stereo sound in the receiver side. Finally, with both PS and SBR, the bitrate is reduced even more, to up to 24 Kbps for stereo audio. Figure 4 presents the encoding and decoding processes for HE-AACv2 profile. The audio stream enters the encoder and reaches first the Parametric Stereo step. The set of coefficients are extracted, injected as ancillary data and the residual enters both the SBR encoder and the downsampler. The SBR coefficients are

Figure 4 – Block diagram of HE-AACv2 profile encoding and decoding steps.



Source and Copyright: (FRAUNHOFER-GESELLSCHAFT, 2013).

calculated and added to the ancillary data, and the final residual is coded with AAC-LC.

LATM is the selected method for encapsulating audio frames for transmission in SBTVD Transport Streams. Further information can be obtained in ISO/IEC 14496-3 International Standard - Audio.

3 ISO/IEC 13818-1 Standard

The main objective of the ISO/IEC13818-1 standard is to describe the conversion of multiple bitstreams into one single stream carrying wrapped video and audio data, as well as additional information responsible to unwrap the streams. Similarly to other ISO/IEC standards, the text only specifies the decoding/demultiplexing side of the transmission chain, leaving the architectures of encoding/multiplexing stages to the manufacturers decision.

Some concepts should be briefly described to the reader by now. The first is the *program* (or *service*), which is formed by a collection of program elements, that on their turn are called *elementary streams*. They may share a time base and then will be reproduced simultaneously. This is equivalent to the analog TV channel concept, in which a broadcaster sends a video stream and several audio streams to the viewer. The second is the *PID*, which is a label assigned at the system layer to each different payload carried by the TS. Each information table has its own PID, each Elementary Stream also. The concept here is similar to a pointer, in programming: during reception of a TS packet, the demultiplexer will read the packet PID and push its payload to the appropriate buffer so that information gets concatenated with previously received packets of the same PID.

The standard defines two architectures that should be applied in different situations, both of them based on the transmission of packets of data. The first is the Program Stream, defined to be used in error free situations, such as storage in digital discs: the streams of this type are composed of long packets of variable length, the Packetized Elementary Streams (PES) packets. The other architecture, suitable for faulty, error prone transmission channels, is the Transport Stream. The stream packet has fixed length and is much smaller than the PES packets, which makes it easier to error correction codes to ensure the transmission through the faulty channels. Since the second architecture is much more suitable to broadcasting situations, the following sections will be focused only on it. Only selected topics will be discussed in the following chapter, these are the concepts that must be understood to develop the simple, but functional, MPEG-2 multiplexer of the project.

3.1 Transport Stream Header

The main structure of the MPEG2 Transport Stream is a packet of constant size. The stream is made of many packets concatenated. Figure 5 shows the basic elements that are present in one packet, the value under each field is its length in bits. From the picture, it can be seen that a packet is roughly formed with 188 bytes of data. The mandatory header has 4 bytes, starts at the *sync byte* and goes up to the *continuity counter*. The

packet payload may also contain the *Adaptation Field*, a sort of header extension, with additional data that helps to decode and present the streams in sync. Apart from the 4 mandatory bytes of the header, the other 184 bytes may be filled with combinations of the following content: the Adaptation Field, PES data or information tables. The Adaptation Field presence is optional, and it is commonly used to carry sync data and to fill empty space in the last TS packets from one PES packet with stuffing bits.

The header fields relevant to this project are briefly described next:

- The *sync_byte* is a fixed, 8-bit field whose value is always `0x47` and is used to identify the beginning of a TS packet;
- The *payload_unit_start_indicator* is a 1-bit field and a value of '1' indicates that either a PES packet or a PSI section starts in the current TS packet;
- The field *PID* is a 13-bit field, that indicates which type of data is stored in the packet payload. As will be discussed later, some PID values are reserved for specific data types;
- The *adaptation_field_control* flag is a 2-bit field and indicates if the current packet mandatory header is followed by an Adaptation Field and/or payload data. Table 1 gives detailed information about the values;
- The *data_bytes* are the packet payload, and shall be contiguous bytes of data from the PES packets, PSI sections, packet stuffing bytes after PSI sections, or private data as indicated by the PID.

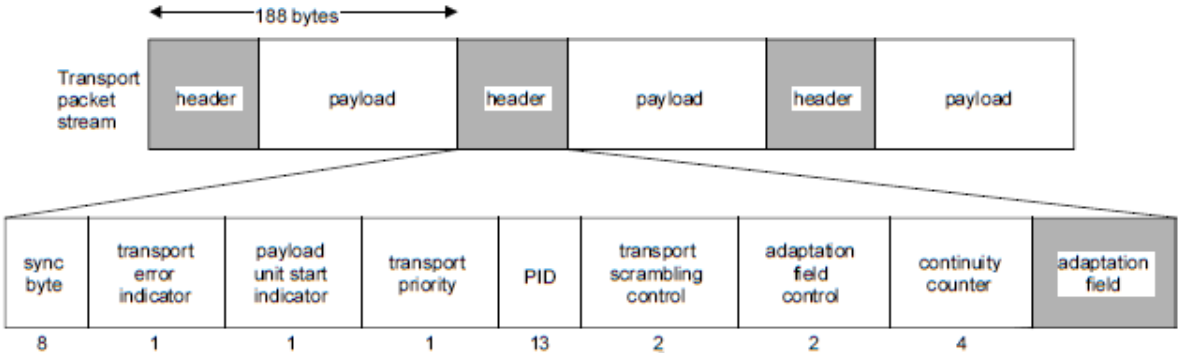
The *continuity_counter* is a 4-bit counter that increments on every TS packet of each individual PIDs. It works as a sequential checking and reordering information for the demultiplexer to know whether the packet order was respected in transmission, or to know if a packet was repeated. Under two conditions the continuity counter is not incremented: if a duplicate packet with payload is sent, and the *adaptation_field_control* must be either '01' or '11' to indicate the payload presence; if there is no payload in the packet, and the *adaptation_field_control* must be either '00' or '10'.

The priority indicator flag and the scrambling control flag are not used in practical implementations and were left out intentionally. However, they are part of the standard and are defined in ISO/IEC (2013, 2.4.3.3).

3.2 Adaptation Field

The Adaptation Field (AF) is an optional extension of the TS packet header. Although optional, it contains many useful information and is widely used in practical

Figure 5 – Formation scheme of a TS packet.



Source: ISO/IEC (2013, F.0.1).

Table 1 – Adaptation Field control values.

Value	Description
00	Reserved for future use
01	No adaptation_field, payload only
10	Adaptation_field only, no payload
11	Adaptation_field followed by payload

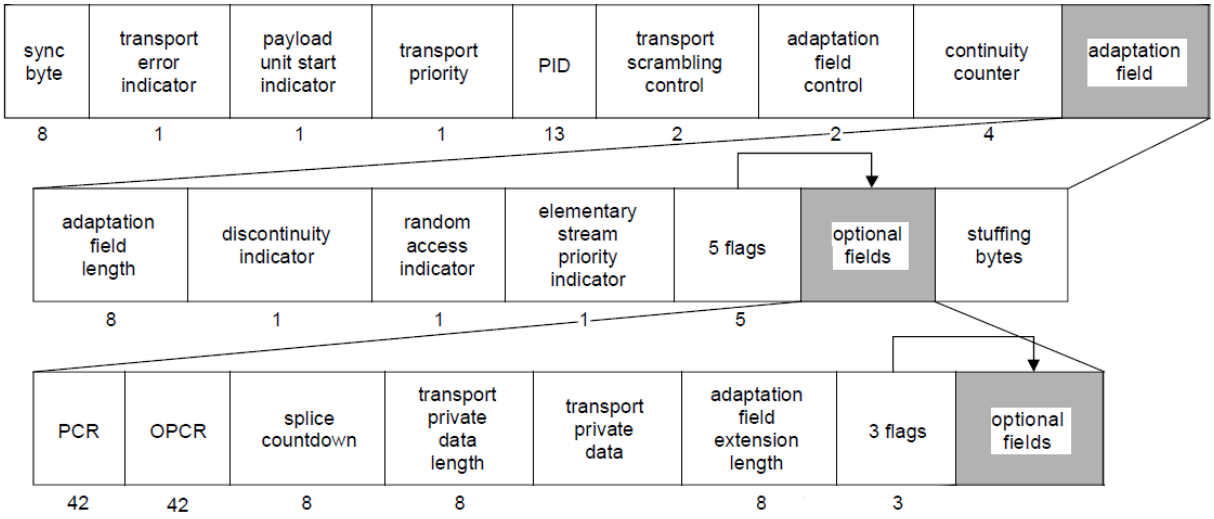
Source: ISO/IEC (2013, 2.4.3.3)

implementations to carry timing information and to fill with stuffing bytes remaining space in TS packets. Figure 6 shows the concatenation of fields in the Adaptation Field graphically. One characteristic of this extension is that many fields are optional, hence there are several flags that toggle the existence of the fields.

The *adaptation_field_length* field has 8 bits and specifies the number of bytes in the adaptation field immediately after itself. When there is payload data sharing the TS packet with the adaptation field, this value shall be in the range 0 to 182. If there is no payload, just the AF, the length must be set to 183. For TS packets carrying PES data, stuffing is usually needed in the last TS packet used because PES packets are not necessarily multiples of 188 bytes. Stuffing is accomplished by setting the adaptation field length to more than the sum of the lengths of the data elements in it, so that the remaining payload bytes in PES data exactly fits in the left space. The extra space in the adaptation field is filled with stuffing bytes with the value `0xFF`.

The *PCR_flag* informs the presence of the optional PCR timing information in the *program_clock_reference_base* and *program_clock_reference_extension* fields, that are described in section 3.4. These fields are wrapped together in Figure 6 as 'PCR', with 42 bits.The other fields will not be described because they are not crucial for the functioning of the multiplexer. However, they are part of the standard and are defined in ISO/IEC (2013, 2.4.3.5).

Figure 6 – Graphical representation of the Adaptation Field.



Source and Copyright: ISO/IEC (2013, F.0.1).

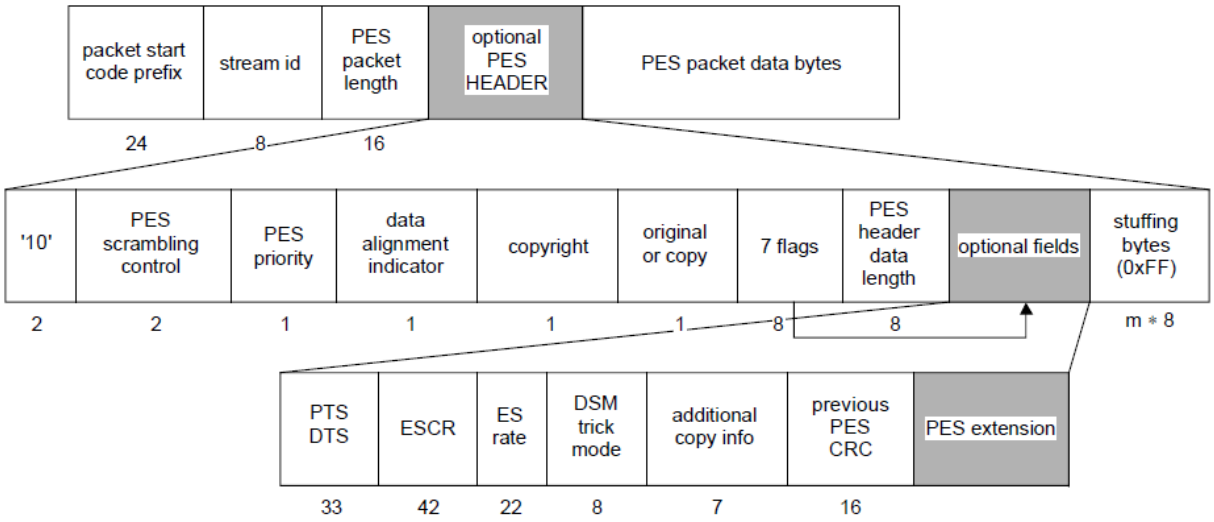
The ISO13818 standard recommends the TS bit-rate to be constant, according to ISO/IEC (2013, 2.4.2.2). However, to increase the performance of the codecs, their output is necessarily variable, and therefore stuffing bits must be added to the transport streams whenever there is no useful information to be transmitted. To better understand this, take the video encoder as an example: consider that the frame rate is constant and equal to 29.97 frames per second. Since the frames are coded according to the H.264 standard with motion compensation and prediction techniques, the I-frames have necessarily more information than P-frames or even more than B-frames. Therefore, during the transmission of an I-frame, the amount of useful information in the output stream is higher than if a B-frame is being processed. It is up to the multiplexer to handle this variable to constant transformation by filling streams with stuffing bytes.

3.3 Packetized Elementary Stream Header

The Elementary Streams, when packetized, also receive a header with Systems Layer information. Figure 7 shows graphically the fields that form the PES Header, and the most significant ones are described in the following list:

- The *packet_start_code_prefix* field has 24 bits and constitutes an identification of the start of a packet. Its value is fixed in `0x000001`;
- The *stream_ID* is a 8-bit field that may be set to a valid value among those in Table 9;
- The *PES_packet_length* has 16 bits and informs the length of the PES packet from the first bit after the field itself until the end of the packet;

Figure 7 – Graphical representation of the PES Header.



Source and Copyright: ISO/IEC (2013, F.0.2).

- Among the *7 flags* field in the second row of Figure 7 is the *PTS_DTS_flags*, a 2-bit field which informs whether time stamps are present for this packet, and whose values are coded according to Table 2. Time stamps are discussed in subsection 3.4.2. These time stamps are carried in the *PTS* and/or *DTS* 33-bit fields, shown in the third row of Figure 7.

Most of the other fields are intended to carry information to help stream decoding, and are out of the scope of this project.

Table 2 – PTS_DTS_flags values.

Value	Timestamp Presence
00	PTS not present; DTS not present
01	Forbidden
10	PTS present; DTS not present
11	PTS present; DTS present

Source: ISO/IEC (2013, 2.4.3.7)

3.4 Timing

Video and audio information are carried in separate streams that need to be reproduced simultaneously in the receiver, so timing control is crucial to the operation of a television broadcasting system.

The timing model introduced by the standard supposes a constant end-to-end delay from the encoder input, through the transmitter, the receiver and up to the decoder output

for all elementary streams of the same program. This way it is possible to theoretically ensure the synchronous playback of both video and audio streams.

The key concept introduced in the ISO standard is the System Time Clock (STC), a reference time base for all the streams in a service being transmitted. The transmitter controls this clock reference for encoding processes and labels each presentation unit with time stamps. The reference clock is sent through the transmission channel and is used by the receiver to lock its own clock to the reference. The media presentation units are transmitted with the time stamps, which are used by the receiver to decode and present the frames at the appropriate time.

If the frequency of the receiver is the same of the transmitter, the encoding, decoding and presentation rates will be the same and the system will work indefinitely as long as there is data to be transmitted. On the other hand, if frequencies are not locked to each other, an inevitable situation of underflow or overflow of the decoder buffers will eventually happen. If an overflow happens, frames will be discarded and not presented. In case of video, if discarded frames would be used to decode other frames, the whole group of pictures is lost. If an underflow happens, there will be freezes in video or mute moments in audio.

It is therefore important to ensure the reconstruction of the STC in the receiver, because the constant delay model is constructed supposing that both sides are perfectly in sync and sets decoding and presentation time stamps to follow the STC. To keep the receiver locked to the reference clock, a digital phase locked loop block is necessary. The following paragraphs describe in greater detail the creation of PCR, PTS and DTS.

3.4.1 PCR

The STC is implemented as a counter with 42 bits incremented at a 27MHz rate. The reference to this oscillator should be considered to be a GPS satellite signal or an atomic clock oscillator, because the frequency deviation may not be above 810 Hz, with a drift smaller than $75 * 10^{-3}$ Hz/s. The STC value is sampled at most every 100ms ISO/IEC (2013, D.0.3) and is encoded in two Program Clock Reference fields (PCR), one in terms of units of 90 KHz and the other in parts of 27MHz. The PCR fields are carried by the Transport Stream inside the TS adaptation field, and might be placed along with Elementary Streams or in individual PIDs. These requirements are defined by ISO/IEC (2013, 2.4).

In the decoder, a digital PLL (Phase Locked Loop) synchronizes the local time reference to the one in the encoder side. The digital PLL is composed by a voltage controlled oscillator(VCO), a counter, a subtractor and a gain adjustment block, as can be seen in Figure 8. At decoder start-up, the VCO is locally set to oscillate at 27MHz and

its signal is used as clock to the counter. Once the first PCR arrives from the received TS, the subtractor outputs the difference between the local PCR and the one sent by the encoder. The result is used as feedback to the VCO, after proper gain adjustment, to sync the local frequency with the encoder. By using this method, the decoder does not need to have an accurate, expensive oscillator, as does the encoder.

The PCR value is not an absolute time reference, but rather the STC starts its counter when the encoding system is turned on. The relationship between the time elapsed since system start-up, $t(i)$, and the PCR value, $PCR(i)$ is calculated as follows, in Equation 3.1. According to ISO/IEC (2013, 2.4.3.5) the PCR is carried in two fields within the TS header. The fields names are program_clock_reference_base and program _clock_reference_extension The first part, program_clock_reference_base, is a 33-bit field whose value is given by PCR_base(i), as given in Equation 3.2. The second part, program_clock_reference_extension, is a 9-bit field whose value is given by PCR_ext(i), as given in Equation 3.3. Yet, as states the standard,

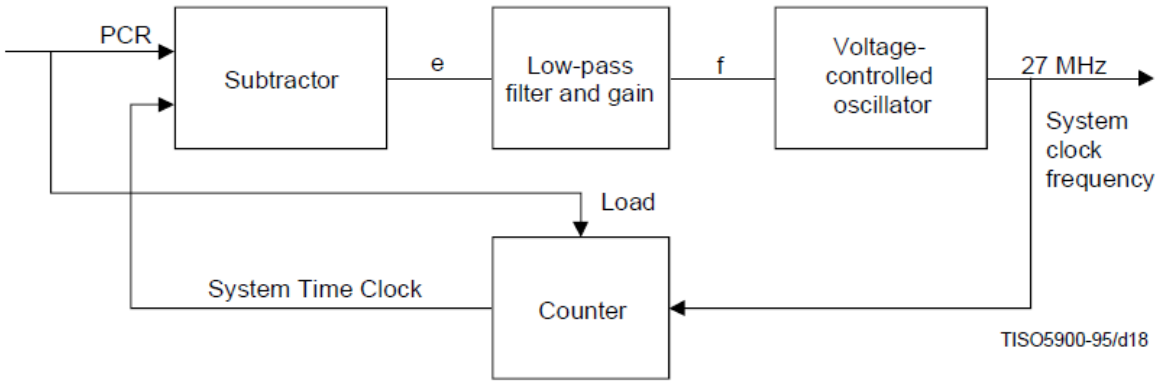
> *the encoded PCR indicates the intended time of arrival of the byte containing the last bit of the program_clock_reference_base at the input of the system target decoder.*

$$PCR(i) = PCR\_base(i) * 300 + PCR\_ext(i) \tag{3.1}$$

$$PCR\_base(i) = ((system\_clock\_frequency * t(i))DIV300)\%2^{33} \tag{3.2}$$

$$PCR\_ext(i) = ((system\_clock\_frequency * t(i))DIV1)\%300 \tag{3.3}$$

Figure 8 – Decoder PLL block diagram.



Source and Copyright: (ISO/IEC, 2013)

Since transmission channels are not perfect, PCR arrival period may vary from one packet to the next, thus subjecting the PCR to jitter. Jitter occurs because packets may arrive out of order or be lost due to transmission errors.

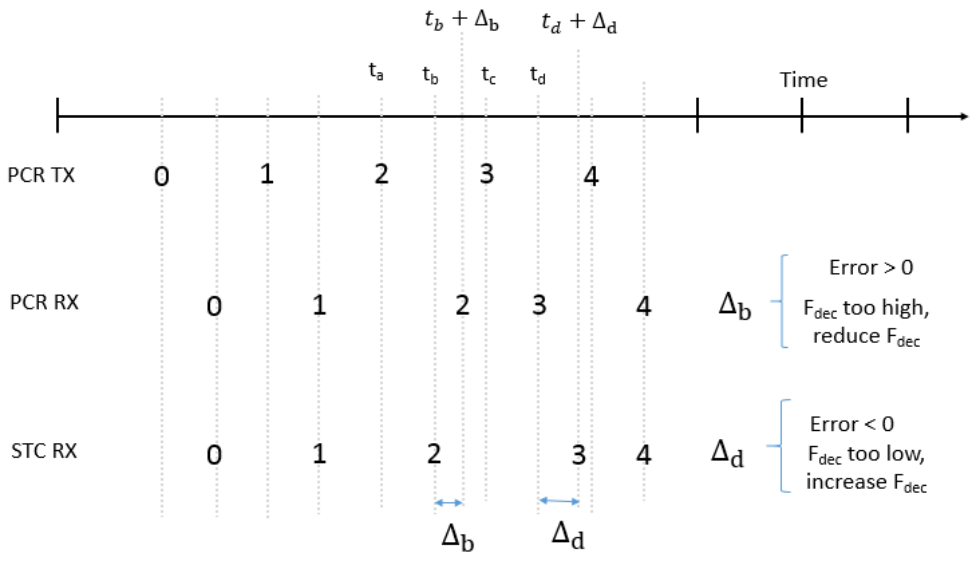
In this situation, a discrepancy between the time of arrival of a given PCR and the time that it represents causes the receiver to lose sync with the transmitter.

Figure 9 shows the consequence in the decoder frequency due to jitter in PCR values. PCR TX represents the times that the PCR values are transmitted. PCR RX represent the instants that PCR values are received. STC RX shows the regenerated clock in the receiver. The values `0` and `1` are sent and received in schedule and a constant delay of half a period can be identified. PCR value `2` is sent at t_a but instead of arriving at t_b, when it was expected, it arrives at $t_b + \Delta_b$. The receiver STC was in sync with the transmitter and generated value 2 at t_b, but since no PCR was received until $t_b + \Delta_b$, a positive error of Δ_b was calculated and interpreted as if the recevier frequency was too high, i.e., the receiver reached PCR value 2 before the transmitter did. Therefore, the frequency is reduced.

Meanwhile, the transmitter sends PCR value 3 at t_c, the packet suffers no jitter and arrives at t_d to the receiver. However, due to the recalculated PLL frequency, the receiver STC will only reach PCR value 3 at $t_d + \Delta_d$, thus creating a negative error value at the subtractor. A negative error value indicates that the internal receiver frequency is lower than necessary and the frequency is increased again. By the time that PCR value 4 reaches the receiver, it is already back in sync and no error is generated.

These oscillations of frequency can be hazardous to program playout, because the receiver buffers are not infinite and a big variation of frequency may lead to buffer overflow or underflow. The consequences of these two situations have already been pointed, they are the loss of groups of pictures or freezes in reproduction.

Figure 9 – Jitter consequence in decoder PLL.



Source and Copyright: The author.

3.4.2 PTS / DTS

During the encoding process, video and audio frames are labelled with Presentation and Decoding Time Stamps (PTS and DTS, respectively). The PTS carries the time that the video and audio access units should be displayed in the decoder output. The DTS determines when the frame should be decoded. As was already explained, video encoders with inter-frame prediction reorder frames during encoding process to reduce the amount of transmitted data. In the decoder, frames may arrive out of presentation order and the decoder uses them to resync frame sequence and ensure the expected presentation frame rate. PTS and DTS are generated in a counter driven by a 90KHz clock, obtained by dividing the 27MHz STC signal.

Take Figure 10 as an example: the upper row shows frames as they are encoded and transmitted; the middle row, as they are decoded; and the lower row shows the presentation order. The I frame receives DTS value 'N' and PTS 'N+1', since the frame must be decoded completely before presentation. The next frames in the presentation order are B_1 and B_2 frames, hence they receive PTS 'N+2' and 'N+3', but they can only be decoded after P_1 frame has already been decoded, so this P_1 frame gets DTS 'N+1' and the B frames would get DTSs 'N+2' and 'N+3'. According to the standard, if the PTS is equal to the DTS, there is no need to send both time stamps and B frames only have PTS. The P_1 frame should be presented after the two B frames, so it gets PTS 'N+4'. The process continues for the next B- and P-frames. The PTS field calculation follows what stated in Equation 3.4: the k-th PTS field for a given stream corresponds to the presentation time $t_p(k)$. The decoder displays frames in the output buffer when its presentation counter reaches $PTS(k)$. The calculation of DTSs follows exactly the same equation.

$$PTS(k) = ((system\_clock\_frequency * t_p(k))DIV300)\%2^{33} \tag{3.4}$$

3.5 Program Specific Information

Program Specific Information(PSI) is a set of normative data useful to transmission, demultiplexing and presentation of audio, video and data streams. The PSI set is organized in tables, each with a specific function in the standard. The tables that carry information about which PIDs carry video and audio streams, called PAT (Program Association Table) and PMT (Program Map Table) are mandatory and their syntax is defined by the ISO/IEC13818-1 standard. The Network Information Table is also mandatory according to the

standard, but its syntax is not defined by ISO/IEC. There is another set of tables that are not mandatory according to ISO13818, but ABNT NBR15603 defines as mandatory, as will be seen in chapter 4.

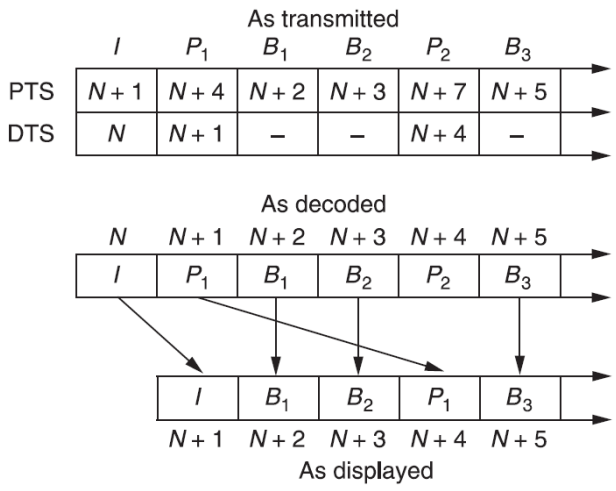
Since programs may change of structure during transmission, for example a movie with multiple audio streams starts playout right after the local news with a single audio stream, PSI tables must follow this change too. To inform the decoder whether a table been transmitted is valid at the moment or not, there are two fields in each section of the tables, the *version_number* and the *current_next_indicator*.

The Conditional Access Table must be present only if scrambling is employed. The Network Information Table is optional and its contents are not specified by ISO/IEC13818-1. The NIT used in SBTVD is defined by ABNT NBR15603.

The ISO standard defines that the PSI tables must be sent in sections and that the sections can be segmented to fit the TS packets. Sections can not be longer than 1024 bytes, although in practice the number of bytes for the mandatory tables is very below that. There are several common fields to all PSI tables, they may vary of value but their syntax is the same:

- The field *table_ID* has 8 bits and defines what is the table's type, according to Figure 37;
- The field *section_syntax_indicator* is a 1-bit field which shall always be set to '1';
- The field *section_length* has 12 bits, the first most significant two shall be '00' and the other 10 bits are the length of the section, starting at the first bit after the length field and including the CRC;

Figure 10 – Example of PTS and DTS use.



Source and Copyright: (WATKINSON, 2004)

- The field *version_number* has 5 bits, and should be incremented by one every time any information in the respective table changes;
- The field *current_next_indicator* has 1 bit, and identifies whether the respective table is valid at the moment of transmission or will be valid later;
- The field *section_number* has 8 bits, and is used to order long, multiple section, tables. In practice, tables usually fit in a single section and this value is set to `0x00`
- The field *last_section_number* has 8 bits, and is used to inform which was the *section_number* of the previous transmitted section. In practice, since tables only fill one section, this field is set to `0x00`, too.

Descriptors are structures used to extend the definitions of services and their elements. All descriptors have the same format: they begin with an 8-bit tag value, then there is an 8-bit length field and variable length data bytes. The standard defines a considerable amount of descriptors, but not all of them are used in practical applications. A selection of the descriptors used in the project are presented in section 4.3.

3.5.1 Program Association Table

The Program Association Table (PAT) indicates the PID of the corresponding Program Map Table (PMT) for each service in the multiplexer. It lists, for every program on the stream, its unique *program number* and the PID of that program's PMT, the *program map PID*. The PAT is assigned to the fixed PID `0x0000`, so the decoder can always know where to look for first and then find the PMT of each service in the PIDs pointed by the PAT. The field *table_ID* is always `0x00` for PAT. The field *transport_stream_ID* has 16 bits and represents a unique ID to differentiate this Transport Stream from all streams in a network. In SBTVD, this value is calculated based on geographic position of the transmitter and the broadcaster's call signs [1]. Additionally to the common fields, this table also carry the following fields, in a loop, right after *last_section_number*:

- first a *program_number* field, with 16 bits, names the first program in the TS;
- then a 13-bit field, *program_map_PID* points the PID value of the corresponding PMT;

[1] A call sign is a unique designation for a transmitting station. In Brazil, the range assigned by ITU for broadcast stations is from ZVA-000 to ZZZ999.

- if the *program_number* is `0x00`, instead of the above definition, ISO specifies that the PID pointed to is the one of the NIT, and the field is called *network_PID*.

A PAT table section is analysed in Appendix D.

3.5.2 Program Map Table

One Transport Stream must have as many Program Map Tables as services. Each Program Map Table (PMT) in the TS indicates which PIDs contain the elementary streams corresponding to one service. It links a **program number** to several **elementary PIDs**. Each elementary stream must have a **type** identification, which designate the nature of the stream, according to ISO13818-1, and defines roughly whether it is a video, audio or data stream. Table 9 is an excerpt of the standardized types. Apart from the common fields already presented in the introductory chapter, the PMT also has the following fields:

- a *program_number* field, with 16 bits, designates the program specified by this section;
- then a 13-bit field, the *PCR_PID* points the PID whose adaptation field contains the PCR that should be used to sync this service ;
- if the *program_number* is `0x00`, instead of the above definition, ISO specifies that the PID pointed to is the one of the NIT, and the field is called *network_PID*.
- The field *program_info_length* has 12 bits, the first most significant two shall be '00' and the other 10 bits represent the length of the section descriptors, in bytes;
- the *stream_type* field has 8 bits, and as already discussed, should be filled with the standardized designation of a stream according to Table 9;
- the *elementary_PID* field has 13 bits and make reference to the PID with the stream whose type is in *stream_type*;
- the *ES_info_length* has 12 bits, the first most significant two shall be '00' and the other 10 bits represent the length of the ES descriptors;
- what follows are a variable number of descriptors, whose syntax will be discussed in section 4.3.

A PMT table section is analysed in Appendix D.

3.5.3 Further Considerations

Although ISO/IEC13818-1 defines other tables, such as the Conditional Access Table, these are not commonly used in free to air television broadcasting systems and were left off the scope of the project.

4
ABNT NBR15603 Standard

4.1 Introduction

The ABNT NBR15603 standard is divided into three parts, each one describes a portion of the SBTVD Systems Layer. Most of the content is an adaptation of the original ISO/IEC13818-1 standard, and there is no reason to be reproduced, but some peculiarities are worth to be detailed. Also, some contents were defined based on the ARIB STD-B10. There is a set of tables not defined in MPEG2 Systems, among them the Network Information Table, that are mandatory in SBTVD. The PIDs that must be used to transmit the Tables are slightly different than what is defined on ISO/IEC13818-1. Figure 38 in Appendix B shows the modified values. Figure 39 also in Appendix B brings other two important information: first, the transmission level of each table, whether its transmission is mandatory(*"obrigatório"* means mandatory) or optional; and the repetition rate of each table in the stream, in units of repetitions per milliseconds. The tables whose transmission is mandatory are Program Association Table (PAT), Program Map Table (PMT), Network Information Table (NIT), Service Description Table (SDT), Time Offset Table (TOT), Event Information Table (EIT) and Conditional Access Table (CAT)(if the transmission is scrambled[1]). The following is a brief description of the mandatory tables that have not yet been discussed in chapter 3.

4.2 System Information

4.2.1 Service Description Table (SDT)

The Service Description Table brings useful information to describe the services contained in the TS. Multiple SDTs can be transmitted, corresponding to different Transport Streams, but in practical broadcast implementations this feature is not used. Services information is coded into SDT in a loop. For every service, it carries flags signalling the presence of another mandatory table, the

[1] Scrambling methods are used to prevent receptors without permission to open certain streams from viewing them. They are usually used in paid television systems but not in broadcast systems.

EIT, which is described in subsection 4.2.4, a *running_status* flag with 3 bits informs to the decoder if the service is currently running; and other flags as described in ABNT (2007a, 7.2.6.1).

4.2.2 Network Information Table (NIT)

The Network Information Table carries information about the physical organization of the network, such as the network ID, and all the characteristics about the transmission channel, such as modulation method, frequency and channels. Most of the useful information of the NIT is carried in descriptors, discussed in subsection 4.3.2. The fields in the NIT section apart from the common fields are the following:

- *network_ID* field, with 16 bits, contains a unique network identification, calculated as described in subsection 4.3.4;
- *network_descriptors_length* contains the byte number of all the network descriptors;
- *transport_stream_loop_length* contains the byte number of all the transport stream loop descriptors;
- *transport_stream_ID* contains the unique TS ID, the same that is in PAT;
- *original_network_ID* contains the same value of *network_ID* field;
- *transport_descriptors_length* contains the byte number of all the transport descriptors.

4.2.3 Time Offset Table (TOT)

The Time Offset Table contains a `UTC-3_time` field, with 40 bits, to hold the information about the current time and date. The two most significant bytes correspond to the least significant bits of the Modified Julian Date ABNT (2007a, Annex A). The other 3 bytes represent the time of the day with 6 digits in BCD encoding, so that 14:54:22 is encoded as `0x145422` in hexadecimal.

The only descriptor that the TOT contains is the *local_time_offset_descriptor*. It indicates the time and date when there will be a transition of time, such as Daylight Saving Time periods and must be compliant to the EN 300 468:2007 standard, subsection 5.2.19 ABNT (2007a, 8.3.25).

4.2.4 Event Information Table (EIT)

The Event Information Table was not yet investigated and is currently not implemented. Although it is mandatory, its absence does not affect the

functioning of the system, as will be shown in subsection 7.3.2.

4.3 Program and System Descriptors

Descriptors are structures of data with a standardized syntax that increase the flexibility of tables to carry specific information related to the broadcasting system, the streams contents and program guides, for example. Their syntax always starts with a *tag* byte, followed by one *length* byte or two, and end with the descriptor *data* itself. The lengths of data fields are flexible, it depends on the information carried by the descriptor. If it is the descriptor of the name of the broadcaster, then it must be variable, but if it's rather the descriptor carrying the central frequency of the channel, it must have always the same size.

The ABNT standard defines a set of descriptors whose presence is mandatory in each of the mandatory tables. They are all listed because one target of the project is to implement all in the application:

- in the PAT there are no defined descriptors;
- in the PMT the following are mandatory: *AAC descriptor* and *Parental rating descriptor*, other descriptors are only mandatory under circumstances ABNT (2007a, 8.1);
- in the NIT the following are mandatory: *Network name descriptor*, *Service list descriptor*, *TS information descriptor*, *Terrestrial delivery system descriptor*, *Partial reception descriptor* (for 1-Segment reception) and *System management descriptor*;
- in the SDT the following is mandatory: *Service descriptor*;
- in the TOT the following is mandatory: *Local time offset descriptor*;
- in the EIT the following is mandatory: *Parental rating descriptor*.

Each descriptor is briefly described in the following sections.

4.3.1 Program Map Table Descriptors

The *AAC descriptor* is not defined in the ABNT standard. The ABNT Operational Guideline for SBTVD suggests the following syntax: the first byte is the descriptor tag, with value `0x7C`; the second byte is the descriptor length, to be filled appropriately; a third byte is to inform the profile and level of the audio streams of the service described by the PMT, whose values are defined in a table(ABNT, 2007b) and are `0x29` for AAC Profile - level 2 or `0x2E` for HE-AAC Profile - level 4, just to cite two examples. The fourth byte has only

a flag to set whether an optional field is used in the following bytes, but it is not used in practical applications.

The *Parental rating descriptor* tag is , after the length byte what follows is a loop of four bytes: the first three represent a country code with three characters encoded according to ISO 8859-15[2], for Brazil the characters are `BRA` which are coded as `0x42 0x52 0x41`. The other byte in the loop is the rating for the service being broadcasted. The first four bits encode the minimum age to watch: from free to all audiences (`'0001'`) to above 18 years only (`'0110'`), with steps in 10 years, 12, 14 and 16. The codes outside this range are forbidden. The other four bits inform the presence of scenes with drugs (`'0001'`), violence (`'0010'`) and sex (`'0100'`). Combinations of these values are possible, so if a show has scenes of drugs and violence, the four bits would be `'0011'`, the most significant bit is reserved for future applications.

4.3.2 Network Information Descriptors

The *Network name descriptor* carries the name of the transmission system. Tag and length have one byte each, and the name is encoded according to the ISO 8859-15 standard in the rest of the descriptor. The network names for two of the local broadcasters in Porto Alegre are "Band HDTV" and "RBS TV POA".

The *Service list descriptor* provides a list of all the services in the Transport Stream. Tag and length have one byte each, and the data fields are formed with a loop of two fields: `service_id` and `service_type`. The service ID is the same as the program number defined in the PMT and explained below. The service type is defined by ABNT ABNT (2007a, 8.3.13): `0x00` for HDTV and SDTV services and `0xC0` for 1-Segment services.

The *TS information descriptor* carries in the `remote_control_key_id` field the virtual channel associated to the transport stream. `ts_name_char` carries the name of the TS, which is the same as in the *network name descriptor*. The field `transmission_type_count` states the number of transmission types in the TS. The field `transmission_type_info` is repeated for each type and is used to designate the hierarchical layers and other transmission parameters. The definition of its value was not found in the standard, so an analysis was done in the local broadcasters streams to discover what were used the values. For HD services, the value is `0x0F` and for 1-Segment it is `0xAF`. Next, a list of the services that are included in each of the layers is added.

[2] ISO 8859 is a character encoding standard similar to the well known ASCII standard, but more complete in terms of characters foreign to the English language. Each character is encoded as a number.

The *Terrestrial delivery system descriptor* indicates several physical conditions of the transmission channel. After tag and length, what follows is an `area_code` field, calculated according to subsection 4.3.4

The *Partial reception descriptor* carries the service ID that contains the 1-Segment reception service. It is only mandatory if partial reception is used.

The *System management descriptor* identifies if the Transport Stream contains free to air or paid content, and if the transmission system is the SBTVD or other. Further information is available in the standard ABNT (2007a, 8.3.21).

The service ID field has 16 bits. The eleven most significant are the less significant on the network ID. The next two bits are the service type, `0x00` for television and `0x11` for 1-Segment television. The last three bits are the service number, a counter representing at most 8 services.

4.3.3 Service Description Table Descriptors

The *Service descriptor* is the only mandatory descriptor in SDT, and there must be as many of it as services in the TS. It carries the name of the provider and the name of each service in the Transport Stream. After tag and length fields there is a service type field, whose value is the same as defined above in subsection 4.3.2. What follows is a sequence of bytes encoding each character of the provider name and the service name.

4.3.4 Special Fields

The area code is calculated as followsABNT (2007a, Annex E). It is formed by 12 bits, the 5 most significant are a unique code per state and the other 7 are an identification of the region according to the IBGE[3]. Next comes the `guard_interval`, with 2 bits mapping the four possible values for the guard interval: 1/32 is `0x00`, 1/16 is `0x01`, 1/8 is `0x10` and 1/4 is `0x11`. Next comes the `transmission_mode`, other two bits mapping the following modes: mode 1 is `0x00`, mode 2 is `0x01` and mode 3 is `0x10`. Finally is added the frequency of the central channel, encoded with the following syntax: $(473+6*(X-14)+1/7)*7 =$ `frequency` in MHz, where X is the channel number from 14 to 69.

The fields `network_ID`, `original_network_ID` and `service_ID` are calculated according to Annex H in (ABNT, 2007a). To calculate the `network_ID` one should take the call sign of the broadcaster, consider only the last letter and the three digits. To each of the possible letters (A, B, P, Q, T) is associated

[3] IBGE is the Brazilian Institute of Geography and Statistics. It is the agency responsible for statistical, geographic, cartographic, geodetic and environmental information in Brazil.

a number from 0 to 4, in the letter order presented here, which is concatenated to the three digits and form a number in decimal base. This number converted to binary forms the `network_ID`. As an example, the standard cites the call sign "ZYB205. The letters "ZY" are discarded and the "B" becomes a 1. The network ID is 1205, that in hexadecimal base is `0x04B5`.

5
Available Solutions and Limitations

A quick search on a search engine for the keywords *MPEG2 TS multiplexer* returns a long list of commercial solutions for multiplexing media files of most different formats in MPEG2 standard. The commercial solutions are usually associated with dedicated hardware, such as the solution presented by (IMMAGINE COMMUNICATIONS, 2014), which are targeted for broadcasters and are expensive, of the order of a few thousand dollars. The main customers of these solutions are television broadcasters, producing live content and need low latency for signal encoding. These devices operate in real time, receiving streams of video, audio and data across multiple ASI interfaces and delivering output in a multiplexed stream in MPEG2 standard. The free solutions are software compatible with Windows and Linux platforms, mostly. Some have graphics and pattern to aid its use by users unfamiliar with the standard interface configurations. The main difference is that the goal is not the user of the system in real time, so that the specification for the processing and memory of personal computers are usually sufficient to run the tools. The input and output interfaces are commonly files that store binary data streams in a sequential manner.

In order to develop the multiplexing tool proposed in the project, some open-source tools were picked to be studied. The criteria for choosing the solutions were the following:

- the application should be freeware and the source code should be under open-source licences;
- the organization of the source code should facilitate its modification to add new features and data;
- the tool should be under current development, solutions without updates in the last 5 years were not considered;
- the tool must contain native compatibility with the elementary streams defined in the SBTVD standard, such as H.264 video and AAC/LATM audio.

A brief description of the solutions that met most of these requirements is presented in the following paragraphs. After that, a tables synthesizes the key

features of the two best of them.

5.1 FFMPEG

According to the developers themselves, FFmpeg is a versatile tool for encoding, decoding, verifying and displaying video, audio and subtitle streams. With a broad set of open source libraries, allows conversion between different formats, frame rate, video frame size, audio sample rate, among other functions. It is the most complete and under most significant development, with daily updates organized in a public repository. Natively, it does not allow the creation of TS streams with multiple services (or programs). It is capable of delivering the output videos encoded in H.264 and AAC-LATM audio. It maintains synchronization between flows, either by keeping the timing information present in the original flows or generating new temporal tags. If desired, a constant TS bit rate can be set and the result is reliable.

5.2 GPAC

The GPAC tool was developed by the French school Télécom ParisTech, that has similar characteristics to FFmpeg, but is less performing in keeping timing accurate and maintaining a constant bit rate of the output file. Supports the creation of multiple services, but demonstrated a random behavior in the number of packets generated for the same input files and parameters. For example, for the same input files with 10 seconds of duration and same input settings, the tool was tested ten times and generated from 7 to 15 seconds of TS output on the ten trials, which suggests a lack of bit rate control.

Table 3 – Comparison of the Available Solutions

Tool	Multi-service	LATM	Sync
FFmpeg	NO	YES	YES
GPAC	YES	NO	NO

Source: Tests made by the author.

5.3 Available Solutions Choice

Even though FFMPEG doesn't provide native support for multiple MPEG services, it was chosen as the base solution to the project because it was noticed

that the modifications to the source code in order to add this feature were likely to be implemented in the available time. The existing timing feature counted towards this option, too. FFMPEG has a public API that could have been used to develop a multiplexing application from scratch, but a modification of the existing structure seemed more rational considering the existing object-oriented architecture and the available development time. Most of the MPEG2 multiplexer tasks are already implemented in the *ffmpeg-formats* utility, though there are critical needs to SBTVD that are not available. The missing features, goals of this project, are the following:

- add support to multiple service MPEG2 transport streams;
- add the tables which are optional in the ISO/IEC standard but are mandatory in SBTVD.

6
Project implementation

6.1 The FFmpeg framework

The source code for the entire FFmpeg application can be downloaded for free from the project website (FFMPEG, 2014). Once downloaded, what the developer encounters is a set of directories and files. In the root directory, there are many text files describing in detail the licences, the configuration, the compilation and the installation of the tool. FFMPEG is organized in the following structure: there are four executable applications that result from the package compilation and installation: *ffmpeg, ffprobe, ffplay* and *ffserver*. These four applications run based on seven libraries that are shared among the applications, and can be also used outside FFMPEG via a public API. The libraries handle the functionalities of the tool, which are basically encoding, decoding, filtering, multiplexing and demultiplexing media streams. The *libavformat* library is responsible for the multiplexing functionalities, it includes a ISO/IEC 13818-1 compatible multiplexer. Even if an application could be developed using the functions in the public API, the development of the original tool is made with object-oriented programming concepts, in a way that interactions between modules are easily understood even for a first-time reader of the code. What follows is a detailed description of the tool, at source-code level. All the code is in C programming language, unless otherwise noted. References for the C language notations used here are widely available on the Internet (CPLUSPLUS.COM, 2014) or on books (KERNIGHAN, 1988).

6.1.1 Variables and Structures

Once the user calls *ffmpeg* with a set of input files, options and an output file, the first step it does is to parse the parameters and initialize the structures that contain streams and system information. The main *libavformat* structure used for both multiplexing and demultiplexing is `AVFormatContext`. When multiplexing, the most important variable this structure carries is an `AVOutputFormat` pointer to the output format handler. It also contains an array of `AVStreams`, which contains all the input streams identified during

parsing step. The class diagram of this structure and its heritage relations is shown in Appendix C.

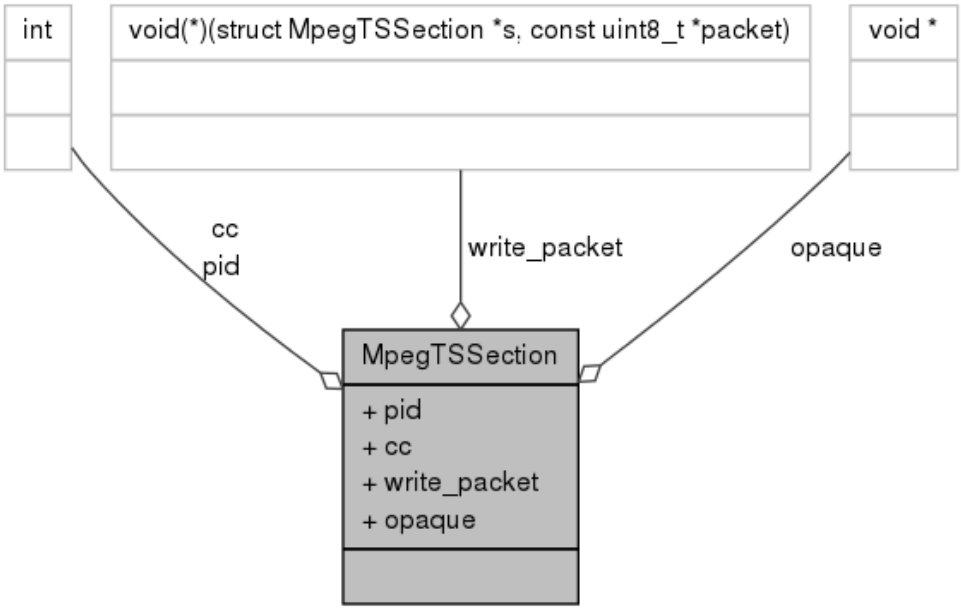
Any *libavformat* multiplexer accepts a set of options, which can be passed as command-line arguments and are checked against a set of preset values during parsing step. While parsing the output file name, if *ffmpeg* identifies a *.ts* extension, the MPEG2 multiplexer inside *libavformat* library is assigned to handle the output. The version number for libavformat at the moment of forking the repository was 55.36.100.

The structures used by the multiplexer that represent the definitions on the standard are `MpegTSSection` for the tables, `MpegTSService` for the services, `MpegTSWrite` for the Transport Stream itself and `MpegTSWriteStream` for the streams.

`MpegTSSection` is the structure that represents one PSI or SI table section and holds information of the table's PID, the current continuity counter status and a pointer to the `AVFormatContext` structure in its `opaque` variable. This is quite a generic section, the customization for each of the PSI tables is done by the functions that write the sections to the stream, which will be commented further in this text. The class diagram of this structure and its heritage relations is shown in Figure 11.

`MpegTSService` represents a service, and contains a `MpegTSSection` for its Program Map Table, a service ID, the PCR PID and a pair of control variables to manage the service's PCR transmission rate. The class diagram of this structure and its heritage relations is shown in Figure 12.

Figure 11 – Class diagram for MpegTSSection structure.

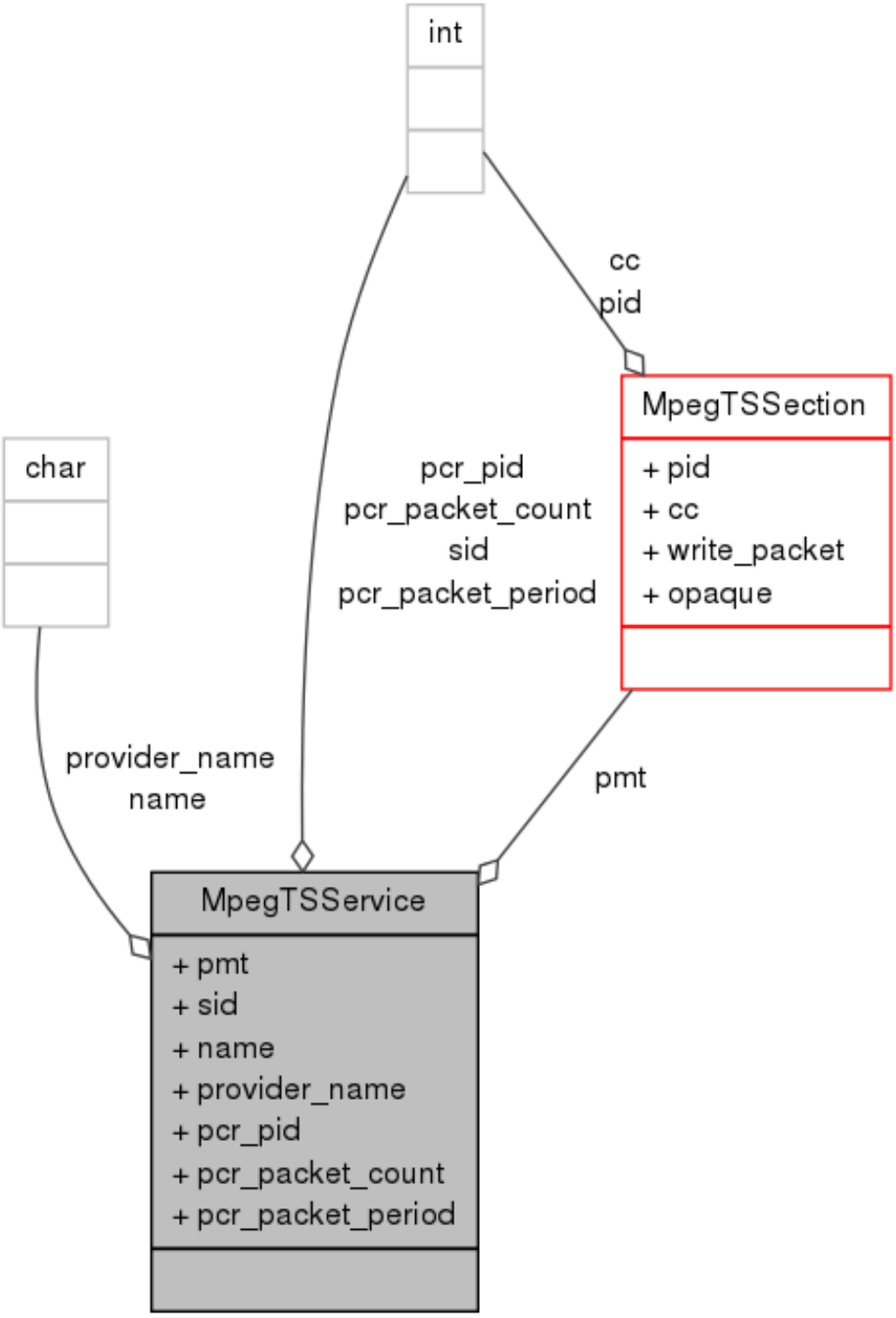


Source: Automatically generated with Doxygen from the source code.

`MpegTSWriteStream` represents one elementary stream that is sent through the TS. It has a pointer to the corresponding service, the stream PID, a continuity counter, the stream's current PTS and DTS values and the stream raw payload itself. The class diagram of this structure and its heritage relations is shown in Figure 13.

The main multiplexer structure is `MpegTSWrite`, shown in Listing B.1,

Figure 12 – Class diagram for MpegTSService structure.

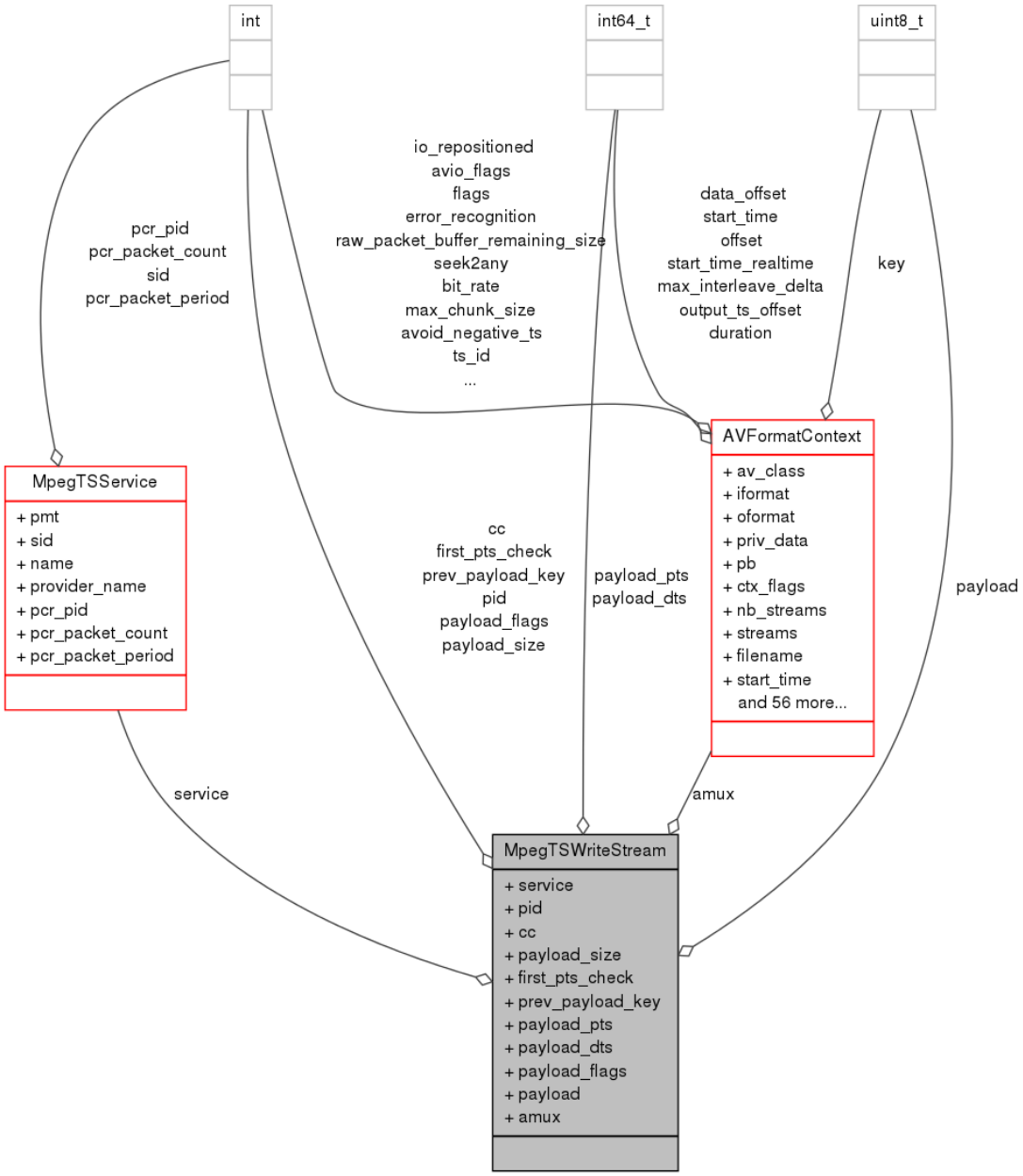


Source: Automatically generated with Doxygen from the source code.

that represents the Transport Stream itself. The structure contains the PSI tables PAT, NIT and SDT as three `MpegTSSection` structures, a dynamic array of `MpegTSServices` and many control variables, such as the counter `nb_services` for the number of services contained in the TS, the desired constant multiplexing rate `mux_rate` and a pair of variables to control the repeated transmission of each PSI table.

Along with the control variables, the physical network information is included as well: the fields `transport_stream_id` and `original_network_id` are assigned values received from the command-line. The class diagram of this

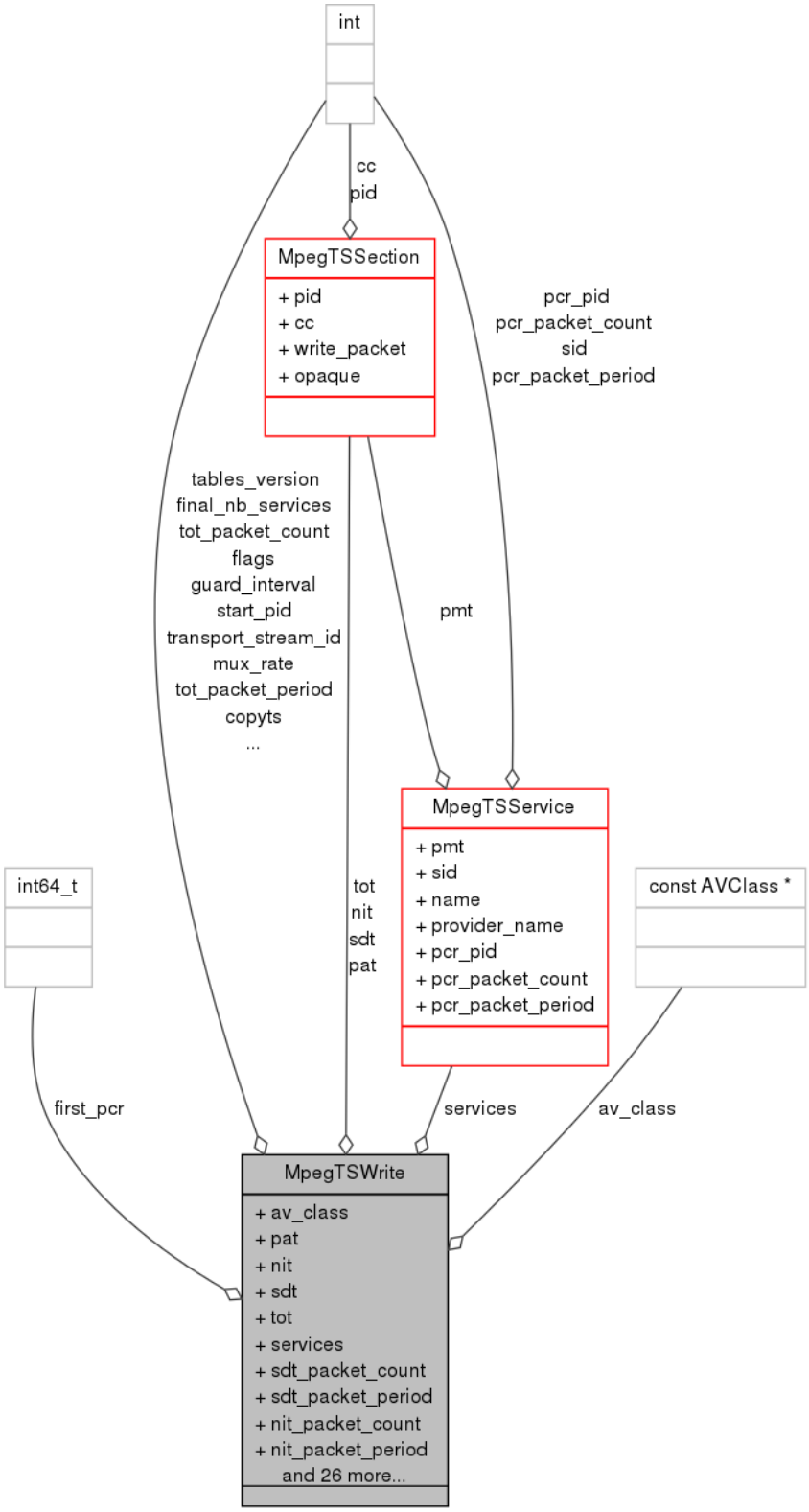
Figure 13 – Class diagram for MpegTSWriteStream structure.



Source: Automatically generated with Doxygen from the source code.

structure and its heritage relations is shown in Figure 14.

Figure 14 – Class diagram for MpegTSWrite structure.



Source: Automatically generated with Doxygen from the source code.

Each command-line parameter passed to FFmpeg is parsed and must be

validated before use. Each valid option is listed as one entry in a structure, an example is presented in Listing B.2. The first field is the option name, the second is a help text, the third is the position within a structure where it should be stored, then comes the variable type, the default value if it is not passed as argument, the minimum value, the maximum, and finally a internal definition of the option.

6.1.2 Functions

The main functions related to the multiplexing process are `av_write_header()` for writing the TS stream header, `av_write_packet()` for writing the packets and `av_write_trailer()` for writting the stream tail. The list below describes the sequence of the calls to the functions:

1. A call to `av_write_header()` initializes the control variables;
2. as long as there is available data in the input streams, successive calls to `av_write_packet()` occur, each one writes one 188-byte TS packet in the output *.ts* file. In its turn, `av_write_packet()` calls `mpegts_write_pes()` to handle the writing process of the current TS packet:
   a) `mpegts_write_pes()` calls `retransmit_si_info()`, that outputs one PSI table section in the packet, if it's time;
   b) `mpegts_write_pes()` loops among the streams and searches a stream with the current frame DTS within the time interval of the current PCR.
3. when input streams are over, `av_write_end()` frees allocated memory and closes the files.

To decide if it's time, in item 2a, `retransmit_si_info()` uses the control variables `*_packet_count` and `*_packet_period` of each table to control the transmission rates by incrementing the corresponding `packet_count` each time the function is called and outputting the table only when the counter reaches `packet_period`;

In item 2b, if `mpegts_write_pes()` finds a suitable PES packet, it fills a TS packet with data from this stream. If not, `mpegts_write_pes()` `mpegts_insert_pcr_only()` is called to output only the necessary for a single PCR, without PES payload nor PSI tables, or `mpegts_insert_null_packet()` is called to output an empty TS packet, full of stuffing bytes.

FFmpeg has dedicated functions to write TS packets with PSI tables. `mpegts_write_section()` is responsible for writing the TS packet header

fields described in section 3.1. `mpegts_write_section1()` is responsible for writing the common fields for all PSI tables, described in section 3.5.

6.2 New input options

In order to provide flexibility to the user to configure the Transport Stream parameters according to his needs, several options were added to the command-line interface.

Most of them materialize the physical characteristics of the network and their names are very related to the definitions in the standards: `mpegts_transport_stream_id`, `mpegts_original_network_id`, `mpegts_area_code`, `mpegts_guard_interval`, `mpegts_transmission_mode`, `mpegts_physical_channel`, `mpegts_virtual_channel`.

Others are used as control variables and are self-explanatory: `mpegts_final_nb_services` and `mpegts_transmission_profile`. Their use is explained in the following sections.

6.3 Multiple services

6.3.1 Introduction

The first proposed modification to the original FFMPEG code is to add multiple service support. The first task was to find a way to give to the multiplexer new information about how many services should be generated. This was done by creating a new command-line parameter: the option `mpegts_transmission_profile`.

Two profiles were designed:

- profile number '1' was designed with two services: one Full-segment service with a high-definition video stream and an audio stream and one 1-Segment service with a low-definition video stream and another audio stream;
- profile number '2' was designed with a variable number of services: up to four services, each with a standard-definition video stream and an audio stream; and one 1-Segment service with a low-definition video stream and another audio stream.

Further information about the Full-Segment and 1-Segment services are provided in Appendix A.

6.3.2 Profile 1: one Full-seg service and one 1-Seg service

First the service ID for the HDTV service is calculated, based on the definitions in subsection 4.3.4: the network ID is the `ts->onid` variable; the service type is hard-coded as `'00'`, as well as the program number (`0x0`). Then, one `MpegTSService` is added to the services array in `MpegTSWrite` with the existent `mpegts_add_service()` function. The parameters to this function are the recent calculated HD service ID, the provider name and the hard-coded service name. The process is repeated for the 1-Segment service, except that the service type is coded as `'11'` and the program number as `0x1`. Finally, the variable `ts->final_nb_services` is set as `'2'`.

6.3.3 Profile 2: standard definition services and one 1-Seg service

Profile two is not yet implemented, but the guidelines are the same as for profile 1. Instead of setting the variable `ts->final_nb_services` as 2, the value is left free for the user to choose and instead of creating only two services, a loop iterates the service creation until a counter reaches the total number of services.

6.3.4 Stream Mapping

When calling FFmpeg, the user defines the order that input streams shall be placed in the `AVFormatContext` structure with the `-map` option. This same sequence is used in the streams loops throughout the code, and defines also the order of assignment of streams PIDs. The first input stream receives the value in `mpegts_pmt_start_pid` and the value is incremented by one for each new stream. The default value for `mpegts_pmt_start_pid` is `0x100`.

As already explained, streams hold the information of which service contains them with a pointer to one `MpegTSService` instance within the `MpegTSWriteStream` structure. Before the modifications of this project, since there was only one service, all the streams belonged to it. Now, the multiplexer iterates through all the streams and does a modulus operation with the stream index and the value of `mpegts_final_nb_services` to assign alternately one stream to each service, as shown in Listing 6.1. `ts_st` is the stream structure, `i` is the iteration variable.

Listing 6.1 – Assigning streams to services.

```
ts_st->service = ts->services[i % ts->final_nb_services];
```

The mapping should be done by placing first the two video streams, then the first audio stream corresponding to the first service, then the audio stream

corresponding to the second service and finally the second audio stream corresponding to the first service.

6.4 PSI tables

Each time `retransmit_si_info()` is called, it updates the counters for each table and tests if they have reached the period value. If true, `retransmit_si_info()` calls the `mpegts_write_<table>()` function, where `<table>` is one of the following: `sdt, nit, tot, pat, pmt`. To add SDT, NIT and TOT support, three sets of conditional calls were added, similar to the one shown in Listing 6.2.

Listing 6.2 – Conditional call to `mpegts_write_nit()`.

```
if (++ts->nit_packet_count == ts->nit_packet_period) {
    ts->nit_packet_count = 0;
    mpegts_write_nit(s);
}
```

A set of macros were added to hold the default values of network characteristics, shown in Listing B.7. In C language, when an array variable is declared, a memory area with the size of the array is allocated and the variable itself is a pointer to the first element of the array Kernighan (1988, 5.1). ISO defines 1024 bytes as the maximum authorized length of a table ISO/IEC (2013, 2.4.4). Excluding 8 bytes of the table header and 4 bytes of the CRC, results in 1012 bytes.

To create the bitstream of the table with pointer manipulation, first an array of type `uint8_t`[1] with 1012 bytes is declared with name `data`. Then a pointer `q` of the same type is declared and the first address of the array is assigned to it.

Since all the bytes in the array are stored in a contiguous memory area, it is easy to create the table bitstream by assigning the byte value of the address pointed by `q` and incrementing the pointer after each assignment, so that the next assignment is done in the next byte of the array. Since section length is dependant on how many bytes are used and length fields are at the beginning of the headers, length fields are skipped in a first pass to be filled later, and an auxiliary pointer(`len_ptr`) holds the position of the length fields for the second pass.

[1] `uint8_t` is an unsigned integer type with size of 1 byte.

6.4.1 Length Calculation

Section lengths are calculated in a simple way. Most of the time their value only counts the variable size part of the sections, from right after the length field and up to the end of the last byte of the section. By calculating the difference of the two pointers `q` and `len_ptr`, one gets the number of bytes written including length bytes. If one subtracts the size of the length field from this difference, he easily obtains the length of the variable portion of the section.

Listing B.8 shows an excerpt of the code: the pointer assignment algorithm. Figure 15 shows an example of how the algorithm works. Six of the bytes allocated to variable `data` are shown on top, empty.

1. The first `data` address is assigned to `q` , as shown by the continuous arrow.

2. The syntax shown assigns the value `0x10` to the address pointed by `q` and increments by one the pointed address. The continuous arrow shows the pointed address after item 2 is completed.

3. `len_ptr` stores the current position of `q`, as indicated by the dashed arrow; the address pointed to by `q` is incremented by 2 units, as indicated by the continuous arrow, to leave space to the length fields.

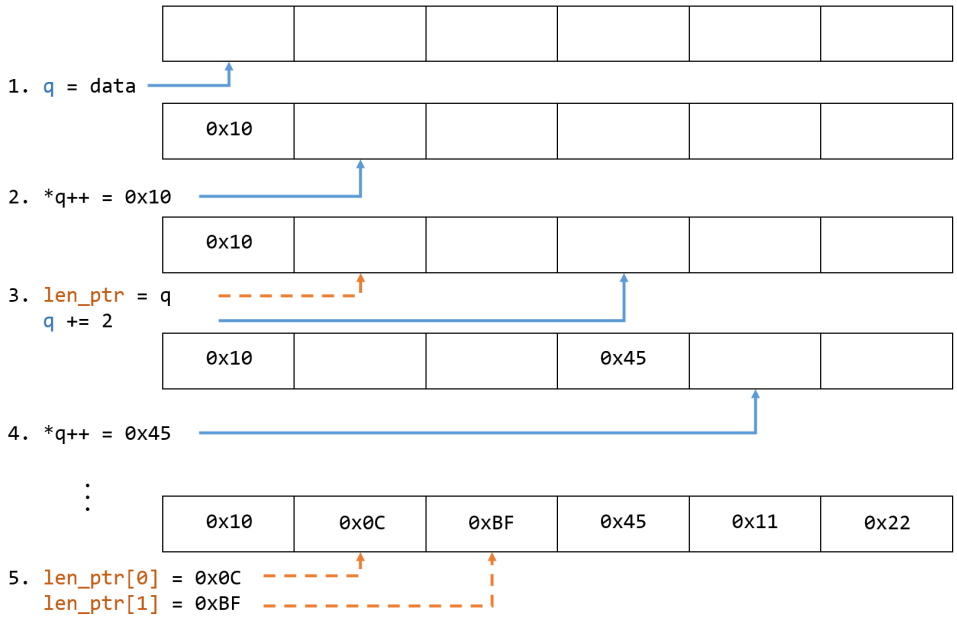
4. the value `0x45` is stored in the address currently pointed by `q`, and `q` is incremented by 1 byte.

5. after all fields are filled with data and the length of the section is known, the length is written using the auxiliary pointer `len_ptr`, that remains pointing to the same position defined in item 3.

6.4.2 Descriptors

In order to facilitate the creation of the descriptors, a prototype of C code was written. It is presented in Listing 6.3. This was possible because the descriptors syntax follow a regular syntax, they always start with a tag byte, followed by a length field with one or two bytes, and followed by data bytes. The calculation of descriptors lengths use the same algorithm described in subsection 6.4.1. All the descriptors in tables from PSI/SI which were described in theoretical sections were coded in the application.

Figure 15 – Pointer assignment example.



Source: The author.

Listing 6.3 – Descriptors prototype.

```
  *q++ = 0x;
  _length_ptr = q;
  *q++;
  put16(&q, 0x);
  *q++ = 0x;
_length_ptr[0] = q - _length_ptr - 1;
```

7
Tests and Results

7.1 Test environment

The laboratory disposes of two systems for TV broadcasting: the first is an EiTV Playout Laboratory (EITV, 2014), that comprises a PSI/SI generator, a multiplexer, a modulator and a low power transmitter. It is a commercial solution, which meets the technical requirements of the SBTVD but the source code is not open to contribution. As inputs, one may supply one or more single-service TS streams as binary files containing video and audio ESs and fill data into forms to be added to the PSI and SI tables. It does not allow input TS files to contain multiple services, so to broadcast a multi-service stream one may only use the internal multiplexer.

As output, the multiplexer produces either a single service or a multi service stream, depending on how the input streams were configured, and the TS is sent in real-time to the modulator and the transmitter.This solution is not totally adapted to the project, since it does not allow multiple services in the same input TS. It is useful, on the other hand, to test the timing of audio and video streams.

The other available solution in the Dektec DTA-115 PCI modulator board, which is installed in a PC workstation. This device is much more adapted to the project goals, because it receives a multi-service Transport Stream as binary file and performs the modulation and transmission steps. It's output is a RF connector which is currently connected to a UHF antenna. A complete description of the board is available in the manufacturer's website (DEKTEC, 2014). The control of the board is done using StreamXpress software, provided by the board's manufacturer.

Two visualization devices are available for each service type: a Philips LCD TV and an EiTV Set Top Box for Full-Segment reception and for 1-Segment reception a Lenoxx GPS with DTV and a Samsung Cell Phone with DTV. Additionally, a Pixelview PenTV USB receiver with Full-segment and 1-Segment reception capabilities is used attached to a Linux Workstation. With the USB receiver, Transport Streams are captured and then analysed

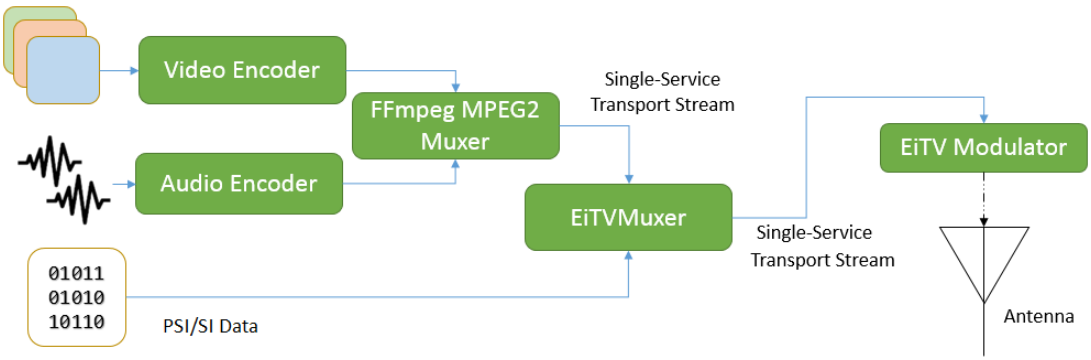
using two free softwares: *MPEG-2 TS Packet Analyser* and *SBTVD Transport Stream Parser v0.32*. FFprobe is also used as a tool to analyse TSs and display information about the streams contained in them.

7.2 Timing tests

These tests were done before any modifications on the source code. The objective of these first tests is to verify that FFmpeg, without any modifications, can provide a stream with video and audio in sync.

Two different sources of video and audio were tested. The first source is a *mp4* file that contains a video encoded in H.264 with resolution of 720x404 pixels, 23.98 frames per second, and audio encoded in HE-AAC with stereo channels at a sample rate of 48KHz. The second source is another *mp4* file with H.264 video, frame size of 480x360 pixels at 29.97 fps, and stereo HE-AAC audio at 48KHz. The two videos were re-encoded with FFmpeg to output a 1920x1080 frame size at 29,97 frames per second, and the MPEG2 multiplexer was used to create a Transport Stream with one single service. The block diagram in Figure 16 shows the signal flow.

Figure 16 – Signal flow for timing tests.



Source: The author.

The syntax to call ffmpeg and produce these outputs is shown in Listing 7.1. `-vcodec libx264 -r 29.97 -s hd1080 -profile:v high` set up the video codec. `-acodec aac -strict -2 -latm 1` set up the audio codec. `-t 60` tells the multiplexer to generate a TS with only 60 seconds.

Listing 7.1 – Single Service TS creation with FFmpeg.

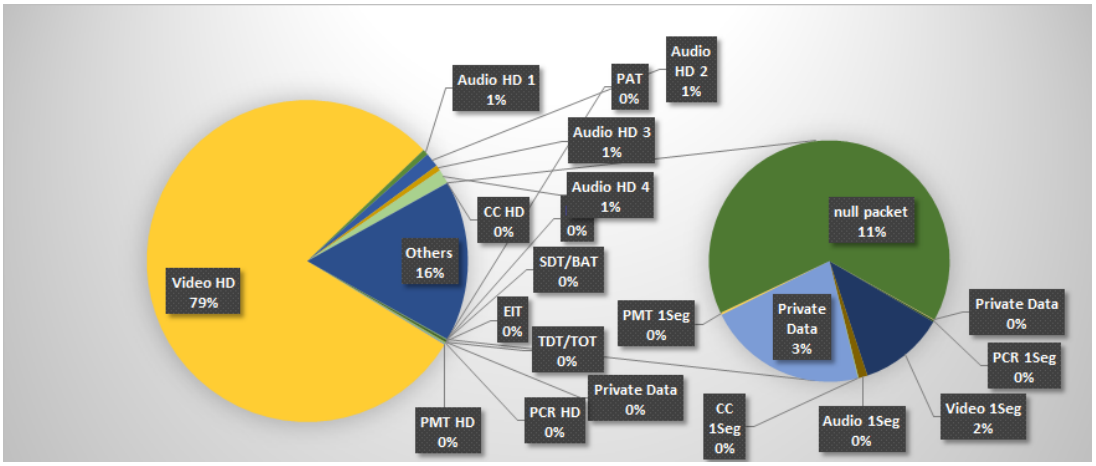
```
./ffmpeg -i src.mp4 -vcodec libx264 -r 29.97 -s hd1080 -
   profile:v high -acodec aac -strict -2 -latm 1 -muxrate
   3600000 -t 60 -mpegts_flags latm -loglevel verbose -y dst.
   ts
```

Attention should be given to the importance of the `muxrate` option. In the first repetitions of this test, FFmpeg was being called without this option and the output TS ended up with a variable bitrate (VBR), which is not compliant to the MPEG2 standard and leads to a catastrophic loss of sync. When setting up the transmission parameters in EiTV control panel, the TS bitrate must be entered, so the average bitrate of the stream was applied. Since a VBR TS was being broadcast as if it had constant bitrate (CBR), whenever the output bitrate was greater than input bitrate, frames were presented faster than supposed to or skipped. On the opposite case, there were gaps without any frame and the audio would mute or the video would freeze.

After realizing the need of setting muxrate, it was necessary to know what value should be set. If an underestimated bit-rate is chosen, the PCRs and PTSs will be calculated incorrectly and frames will not be delivered at the expected frame rate. When the transmitter eventually sends the bit-stream to the air in a slower bitrate than necessary, playback will present freeze moments because there will be instants of time without video or audio to be displayed, i.e., it is as if the stream was reproduced in slow motion. On the other hand, an overestimated TS bit-rate causes the multiplexer to add too much stuffing packets into the stream and use unnecessary band.

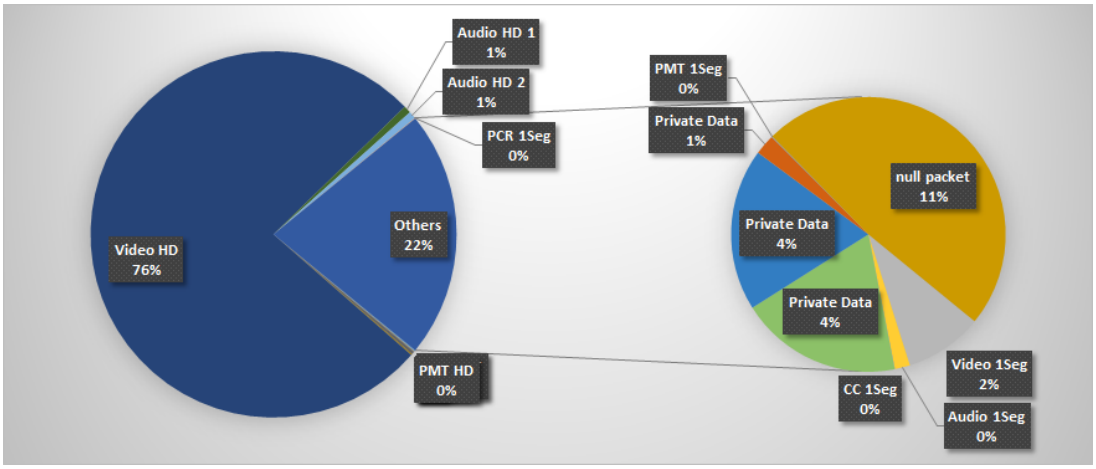
Analyses of the local broadcast channel shown that TSs carried about 11 % of stuffing bytes, as can be shown in the graphics in Figure 17 and Figure 18. The percentages indicated in the figures refer to the amount of data in each PIDs. The PID numbers and packet counts can be seen in Table 4. In the graphs, the designation of stuffing packets is *null packet*. From this information, the muxrate option was set at 3.6 Mbps, which results in an overhead of about 11 %, as can be seen in Figure 19.

Figure 17 – Packets distribution among PIDs in local broadcaster A.



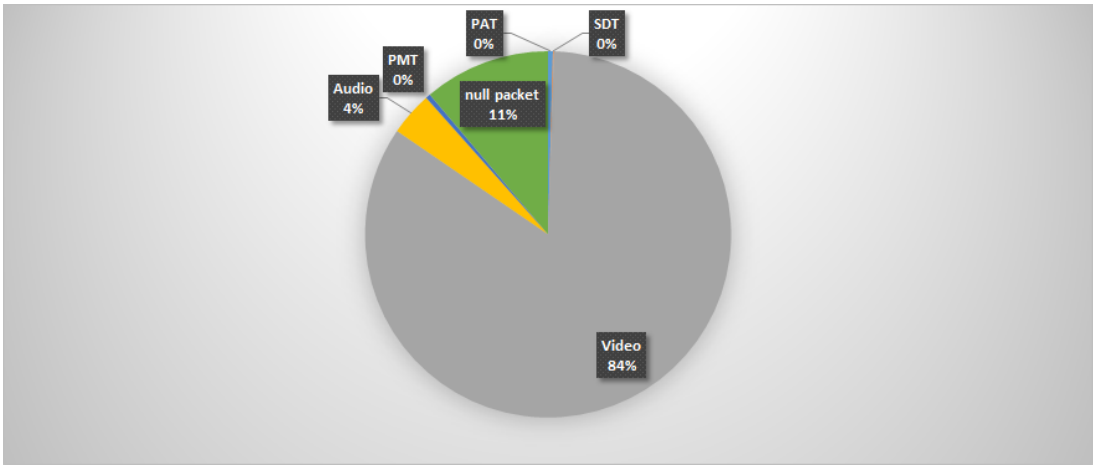
Source: The author.

Figure 18 – Packets distribution among PIDs in local broadcaster B.



Source: The author.

Figure 19 – Packets distribution among PIDs in generated TS.



Source: The author.

Listing 7.2 – Output of FFmpeg when input is as Listing 7.1.

```
Input #0, mpegts, from 'dst.ts':
  Duration: 00:00:59.98, start: 1.445400, bitrate: 3662 kb/s
  Program 1
    Metadata:
      service_name    : Service01
      service_provider: FFmpeg
    Stream #0:0[0x100]: Video: h264 (High) ([27][0][0][0] / 0
       x001B), yuv420p, 1920x1080, 29.97 fps, 29.97 tbr, 90k
       tbn, 59.94 tbc
    Stream #0:1[0x101](und): Audio: aac_latm ([17][0][0][0] /
       0x0011), 48000 Hz, stereo, fltp
```

FFprobe displays what is in Listing 7.2 when analysing the output of Listing 7.1. The TS lasts 59.98 seconds and has a bitrate of 3662 kbps. It contains only one program, with program_number 1, that has two streams:

one video stream (PID `0x100`) and one audio stream (PID `0x101`) that comply to the requirements of the SBTVD standard.

The streams were received in the visualization devices and the sync between video and audio was verified by watching and hearing the outputs in scenes where people were filmed while talking. Three different people stated that video and audio were in sync.

The results were satisfactory, and although there was no employment of functionalities developed by the author, this test was necessary to ensure that the multiplexer could packetize correctly the chunks of data, as well as label the frames with timestamps in sync.

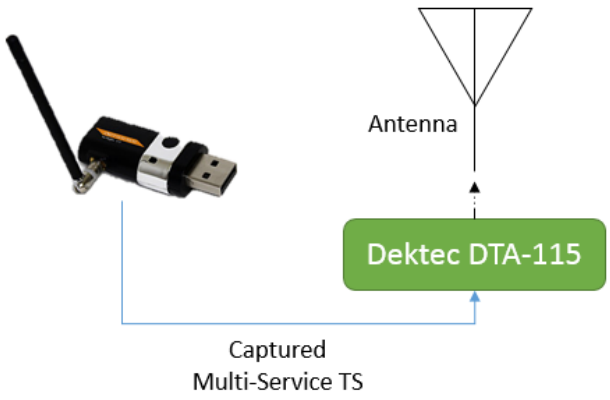
7.3 Multiple Services Tests

These tests were carried out after applying the implementations described in chapter 6. Their purpose is to validate the developed system. Two different sets of tests were made, one with a TS captured from a local broadcast channel with the USB receiver and the other with a TS created with the developed multiplexer.

7.3.1 Retransmitting a captured TS

This first set of tests were carried out to understand how the Dektec PCI board works and to validate its functioning. It was done by transmitting a TS that certainly is compliant to the standard, because it was captured from a local broadcaster. The simple block diagram in Figure 20 shows the signal flow in this test.

Figure 20 – Signal flow for retransmission tests.



Source: The author.

The modulator board was configured with the following configuration, which is known to work as informed by previous researchers of the laboratory:

- television broadcast type, mode 3, guard interval 1/16;
- partial reception enabled for 1-Segment transmission on layer A;
- layer A configured with QPSK modulation, code rate 2/3, time interleave 2 and occupying one segment[1];
- layer B configured with 64QAM modulation, code rate 3/4, time interleave of 2 and occupying twelve segments[1];

With the captured TS file loaded in StreamXpress and these configurations set, in the program interface it is possible to select which PIDs should really be transmitted and in which layer in real-time. Several different combinations of transmitted PIDs and the status of reception in the TV and cell phone are organized in the following tables. They are separated by the reception device and within each device separated by its tuning configuration.

Several important information can be observed from this tables:

- all times PAT and NIT were transmitted in layer A, so it can be noticed from Table 5 that the TV is capable of decoding PIDs sent in the 1-Segment layer;
- during blind scan, the TV can not find the broadcast channel without the NIT as proved by Table 7;
- in Table 6, without the PMT, even if the A/V ESs are sent, the TV can not open the streams;
- the SDT doesn't make difference to the tuning or finding PIDs, but without it the service has no name.

With the cell phone, the results are slightly different. There is no manual tuning method, so only blind searches and normal operation tests were done. The results are:

- from Table 8 it can be seen that PAT presence makes no difference for the phone to open the streams. This is due to the recommendation on NBR 15608, for 1-Segment PMTs to have a default PID range;
- without NIT, the phone can not find the broadcast channel in blind search. If NIT is removed after blind search, the phone keeps receiving as long as PMT is still being sent.

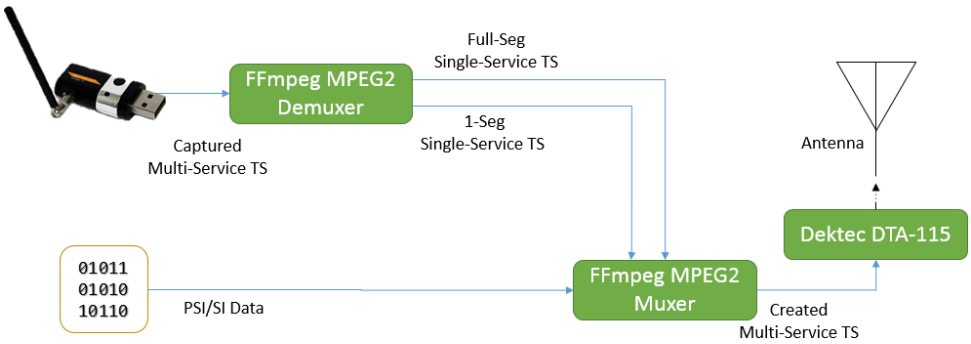
This whole set of tests provides now information about what to expect when the stream generated by the developed multiplexer is broadcasted.

[1] Refer to Appendix A.

7.3.2 Transmitting a remultiplexed TS with two services

The main objective of this last set of tests is to confirm that the developed multiplexer creates a TS compliant to the standard and that it is received and decoded correctly. The two inputs are edited Transport Streams which were captured and remultiplexed, one contains the Full Segment service and the other, the 1-Segment service. The signal flow is shown in Figure 21

Figure 21 – Signal flow for remultiplexed TS tests.



Source: The author.

The TS was generated using the following configurations, which are compliant to the standard: the physical network characteristics are original network ID 0730, area code 2970, guard interval 1/16, transmission mode 3, physical channel 20 and virtual channel 1. The transmission profile is set as 1. Listing 7.3 shows the command-line call and Listing 7.4 shows the FFprobe analysis results.

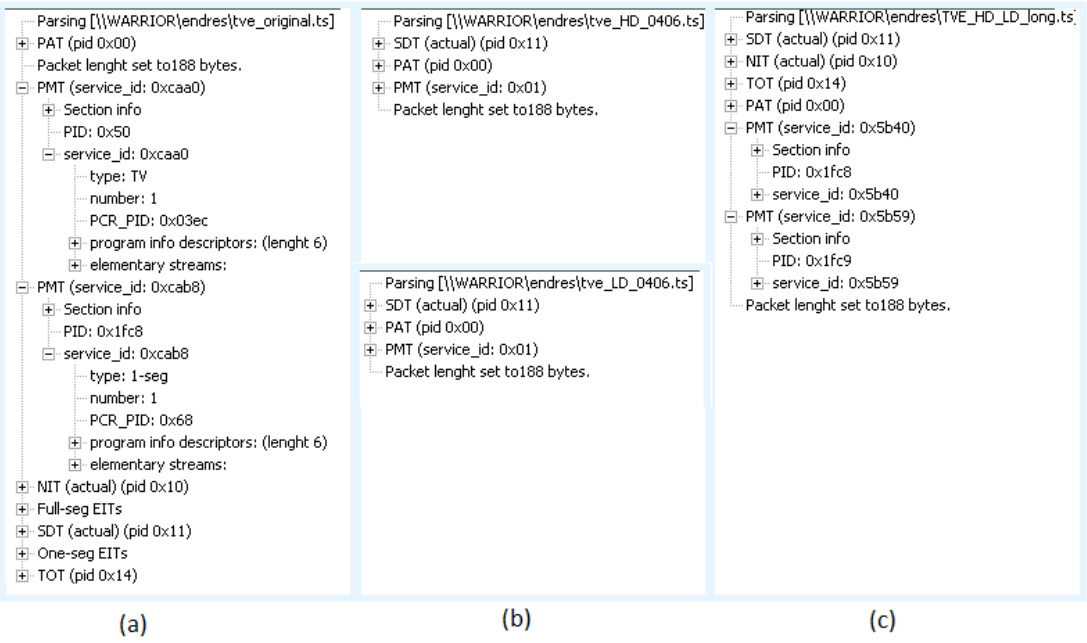
Listing 7.3 – FFmpeg command-line call to generate a multi-service TS

```
./ffmpeg -i /var/src/tve_HD_0406.ts -i /var/src/tve_LD_0406.ts
    -map 0:0 -map 1:0 -map 0:1 -map 1:1 -vcodec copy -acodec
   copy -mpegts_original_network_id 0730 -mpegts_area_code
   2970 -mpegts_guard_interval 2 -mpegts_transmission_mode 3 -
   mpegts_physical_channel 20 -mpegts_virtual_channel 1 -
   mpegts_transmission_profile 1 -muxrate 12000000 -
   mpegts_flags latm -loglevel verbose -t 30 -y /home/nethome/
   endres/TVE_HD_LD.ts
```

Listing 7.4 – FFprobe analysing the multi-service TS.

```
Input #0, mpegts, from '/home/nethome/endres/
   TVE_HD_LD_long_1406.ts':
  Duration: 00:01:01.36, start: 1.400000, bitrate: 11972 kb/s
  Program 23360
    Metadata:
      service_name    : SVC HD Full Seg
      service_provider: FFmpeg
    Stream #0:0[0x100]: Video: h264 (High) ([27][0][0][0] / 0
       x001B), yuv420p(tv, bt709), 1920x1080 [SAR 1:1 DAR
       16:9], 29.97 fps, 29.97 tbr, 90k tbn, 59.94 tbc
    Stream #0:1[0x102](por): Audio: aac_latm ([17][0][0][0] /
       0x0011), 48000 Hz, stereo, fltp
  Program 23385
    Metadata:
      service_name    : SVC LD 1-Seg
      service_provider: FFmpeg
    Stream #0:2[0x101]: Video: h264 (Constrained Baseline)
       ([27][0][0][0] / 0x001B), yuv420p, 320x240 [SAR 1:1 DAR
        4:3], 14.99 fps, 29.97 tbr, 90k tbn, 29.97 tbc
    Stream #0:3[0x103]: Audio: aac_latm ([17][0][0][0] / 0
       x0011), 48000 Hz, stereo, fltp
```

Figure 22 – TS Parser output for four different Transport Streams.



Source: The author.

Figure 22 shows the output of MPEG2 Parser for the original captured TS in (a), the two streams derived from it in (b) and the remultiplexed TS by this project in (c). In (a), one may notice the presence of all PSI/SI tables. In

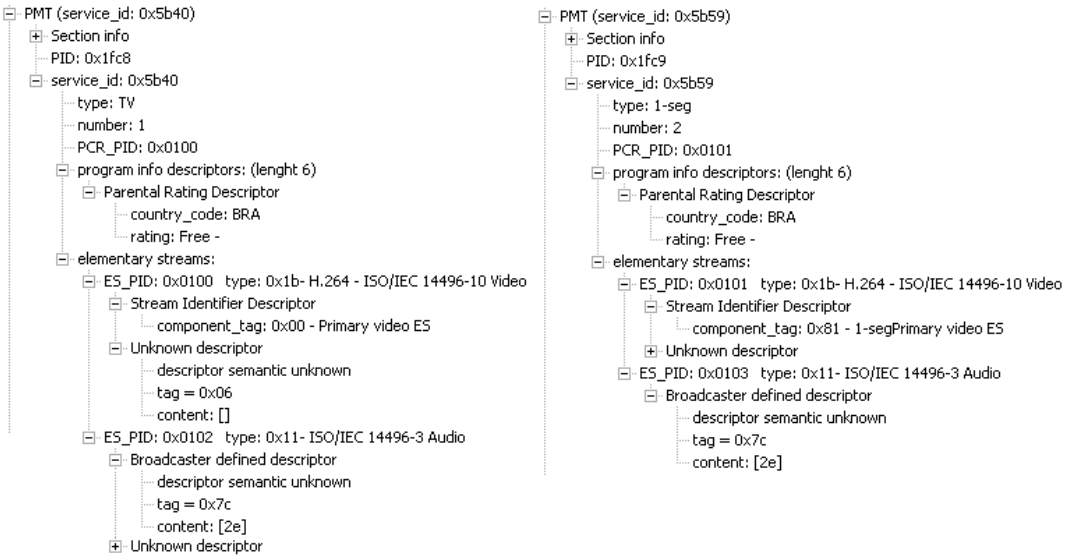
(b), only one PMT remains on each edited TS and since they were generated with the original FFmpeg, SI tables are not present(NIT, TOT), except the SDT[2]. In (c), the PSI/SI tables are recreated with the developed solution and the service numbers are no longer the same, since network characteristics are different from the original captured TS.

In Figure 23 the TS Parser is analysing the PMTs of the remultiplexed TS. In the left side is the PMT for Full-Segment service, where service ID `0x5B40` can be identified, the parental rating descriptor is indicating that the show if free for all ages in Brazil and the Elementary Streams described are one video of type H.264 at PID `0x100` (the primary video ES) and one AAC/LATM audio of PID `0x102`. The parser lacks the syntax of some descriptors and can not show the AAC descriptor, for example, but it can be seen that its tag is `0x7C` and content is `0x2E` as indicated by subsection 4.3.1.

In the right side of Figure 23, the same as above is valid, but the video stream is identified as 1-segment primary video and the service number is `0x5B59`.

It is interesting to show here that the stream mapping explained in subsection 6.3.4 actually worked. The first input stream was the HD video and it received the PID `0x100`. The second was the 1-Segment video, assigned PID `0x101`. The third was HD audio, that got PID `0x102`, and 1-Segment audio got PID `0x103`.

Figure 23 – TS Parser analysing PMTs of remultiplexed TS.



Source: The author.

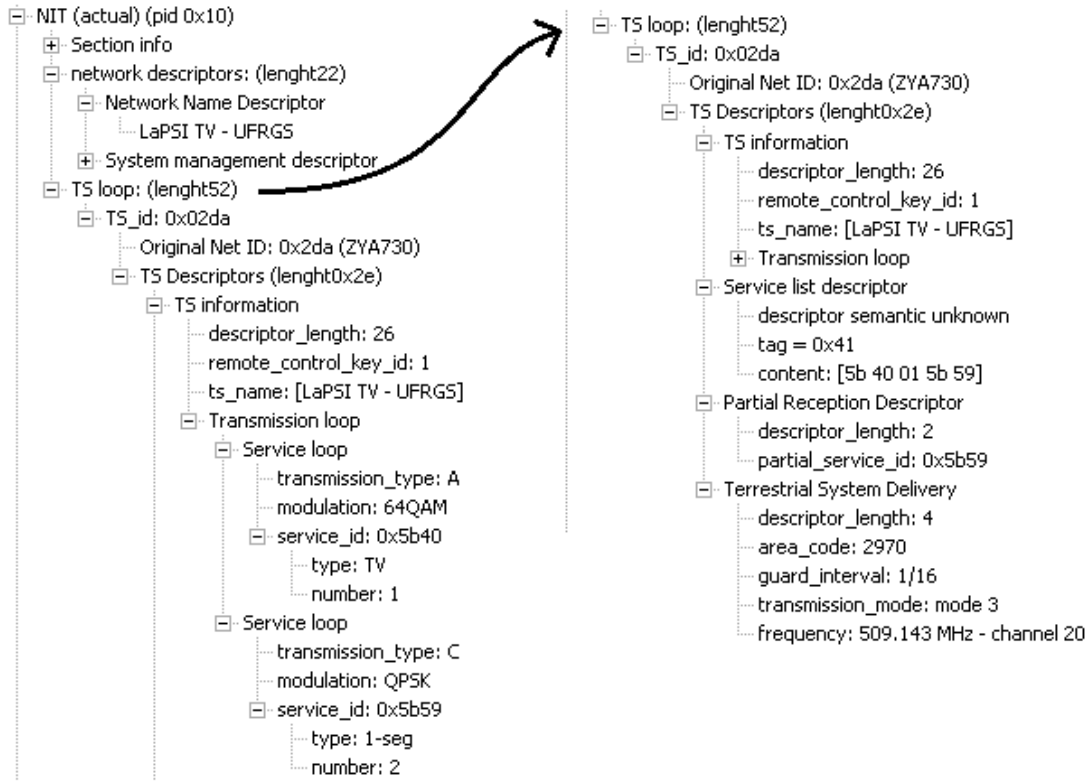
In Figure 24 the TS Parser is analysing the NIT of the remultiplexed TS. Since the table is too long, it was split in two parts as shown by the arrow.

[2] An SDT without all the required information is generated by FFmpeg automatically.

From the network descriptors tree, it is seen that the default network name was used in the Network Name Descriptor. The parser lacks the syntax of the System Management Descriptor. Further in the tree, one sees the value ZYA730 as the Original Network ID, the value that was passed as a parameter to FFmpeg.

Within the TS descriptors, the TS name holds the same name of the network, as defined by the SBTVD standard. In the Transmission loop, the two services have their modulation schemes detailed, as well as the service types, numbers and IDs. In the rightmost part of Figure 24, the Partial Reception Descriptor points that service `0x5B59` is for 1-Segment reception. Finally, the Terrestrial System Delivery Descriptor carries many of the options passed to FFmpeg, all according to the expected.

Figure 24 – TS Parser analysing NIT of remultiplexed TS.



Source: The author.

In Figure 25 the TS Parser is analysing the SDT of the remultiplexed TS. Once again the service IDs and service types are shown, as well as the EIT scheduling flags indicating that there is neither EPG information nor EIT tables. The Service Descriptors show the service names, providers and service types.

After this complete analysis among the PSI/SI tables generated by the code developed in the project, it can be seen that all the transmitted tables and descriptors are compliant to the standard and contain valid information.

Figure 25 – TS Parser analysing SDT of remultiplexed TS.

Source: The author.

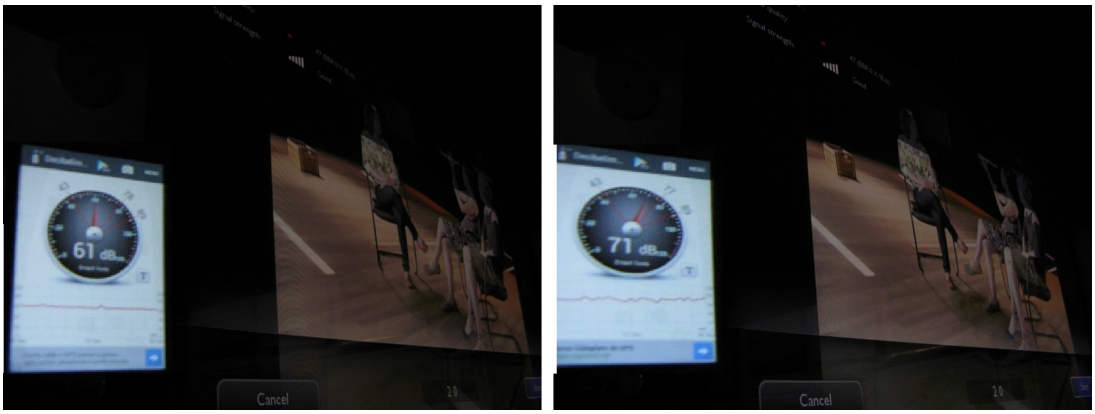
Also, the information in the tables is readable by the receptors as will be seen next. The EIT table and its descriptors, not yet implemented, did not block the system functionalities. The TOT table is currently presenting errors in operation and although it was generated, it was not broadcasted.

The same sequence of tests of subsection 7.3.1 was carried out for this Transport Stream. The results were similar as in the other tests, which is why the tables with results are not presented. Instead, some pictures were taken to illustrate the system working.

Figure 26 shows a scene with a dB meter close to the TV speaker. In the left picture are being transmitted PAT, PMTs, NIT, SDT and video ES, but no audio ES; the audio level is 61dB. In the right picture the audio ES is sent along with the other PIDs and the level increases (to 71dB), as expected.

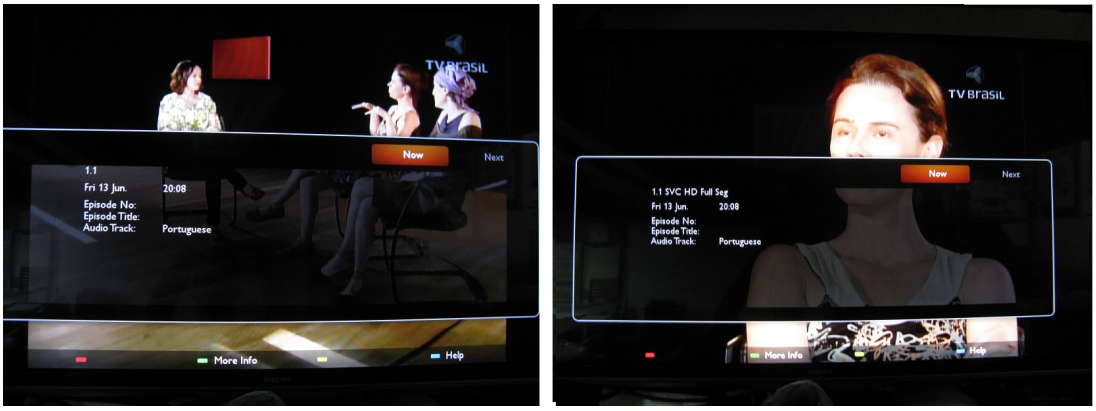
In Figure 27 it can be seen the effect of sending the SDT into the Transport Stream. In the left side, there is PAT, PMT, NIT but no SDT, and therefore there is no name next to the virtual channel number. In the right, on the other hand, the SDT is sent and the name "SVC HD Full Seg" is shown, which is the default name configured by the C macros in FFmpeg, as expected.

In the left side of Figure 28, the reception on the cell phone can be seen. Overlapped to the video is the channel list with the selected virtual channel '1' and the network name describing the channel. In the right side, a manual tuning is being performed in the channel 20 of the TV and no ESs nor the SDT are being sent, only PAT, PMTs and NIT. It can be seen that the channel name is empty but there is a "good" reception level. The "Channels found" flag indicates that PAT and PMT for the HD service are present, but the blue screen points that there is no ESs to be decoded.



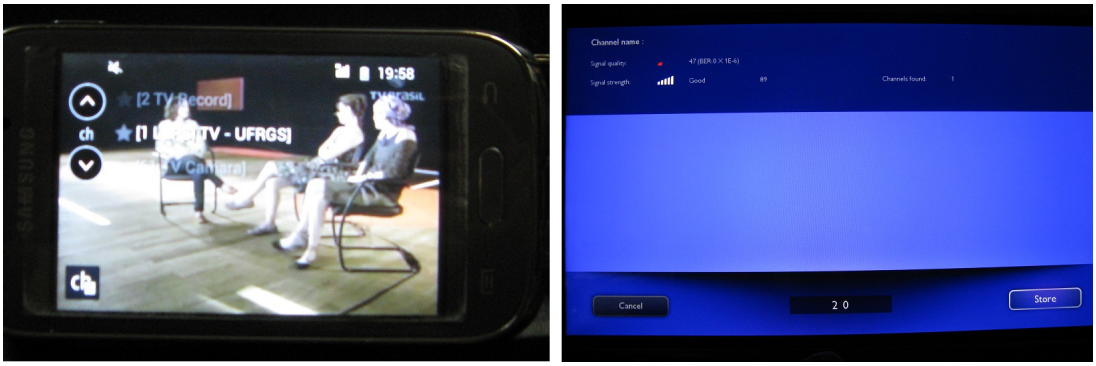
Source: The author.

Figure 26 – Reception of video with and without audio.



Source: The author.

Figure 27 – Service information with and without SDT.



Source: The author.

Figure 28 – Reception in the Cell Phone and TV manual tuning.

Table 4 – Data from stream aquisitions.

Bcaster A	Bcaster A	Bcaster A		Bcaster B	Bcaster B	Bcaster B
PID	Type	Count		PID	Type	Count
0	PAT	260		0	PAT	100
16	NIT	26		16	NIT	10
17	SDT/BAT	13		17	SDT/BAT	5
18	EIT	734		18	EIT	10
20	TDT/TOT	5		20	TDT/TOT	2
39	Private Data	115		36		10
256	PCR HD	452		39		10
257	PMT HD	261		233	CC HD	27
273	Video HD	237551		256	PCR HD	264
274	Audio HD 1	1729		257	PMT HD	100
275	Audio HD 2	4288		273	Video HD	76284
276	Audio HD 3	1745		274	Audio HD 1	654
277	Audio HD 4	4280		275	Audio HD 2	660
278	CC HD	53		512	PCR 1Seg	47
500	Private Data	51		529	Video 1Seg	2006
512	PCR 1Seg	114		530	Audio 1Seg	394
529	Video 1Seg	5551		549	CC 1Seg	4
530	Audio 1Seg	585		1000	Private Data	4169
534	CC 1Seg	53		1001	Private Data	4168
900	Private Data	10415		4101	Private Data	521
8136	PMT 1Seg	132		8136	PMT 1Seg	26
8191	null packet	31587		8191	null packet	10529
Total		300000				100000

Table 5 – TV receiver in manual tuning method.

Sent PIDs	Signal Level	Screen / Speakers Status	Service Name
NIT	0	Blue Screen	NONE
NIT, PAT	Good	Blue Screen	NONE
NIT, PAT, PMT	Good	Blue Screen	NONE
NIT, PAT, PMT, A/V ESs	Good	Video and Audio in Sync	NONE
NIT, PAT, PMT, A/V ESs, SDT	Good	Video and Audio in Sync	HDTV SVC

Table 6 – TV receiver in normal operation mode.

Sent PIDs	Opens A/V Streams	"No Signal" Flag	"No available programme" flag
NIT	NO	YES	NO
NIT, PAT	NO	NO	YES
NIT, PAT, PMT	NO	NO	YES
NIT, PAT, PMT, Video ES	YES, video only	NO	NO
NIT, PAT, A/V ESs	NO	NO	YES
NIT, PAT, PMT, A/V ESs	YES, both	NO	NO

Table 7 – TV receiver in blind search mode.

Sent PIDs	Finds channel
NIT	YES
PAT, PMT	NO
PAT	NO
NIT, PAT	YES

Table 8 – Cell phone in blind search and normal operation.

Sent PIDs	Finds channel in blind search	Status in normal operation
NIT	YES	"Signal too weak, retry?"
PAT, PMT, SDT, A/V ESs	NO	Opens A/V
NIT, PMT, SDT, A/V ESs	YES	Opens A/V
NIT, PMT, A/V ESs	YES	Opens A/V

8
Conclusions and Future Development

Television is the most widespread communication media in Brazil, terrestrial broadcasts cover up to 70% of the population and people rely on television to fulfil their needs of information and entertainment. Differently than Internet, TV has roughly a unidirectional communication channel. With the growing Internet coverage, people are discovering the benefits of interactivity, which is impossible with analog television systems. Interactivity is possible if multiple data can be sent to the receiver, and the options presented to the user. By combining TV coverage to interactivity, social inclusion of remote or poor areas can be promoted. Several projects are already in development, such as the Brasil 4D application that delivers lists of job vacancies and allows users to schedule medical appointments.

Also, paid TV systems deliver high-quality video and audio contents, and subscriptions are becoming cheaper every year. There is a strong risk that, without the migration of free to air analog television to a high-quality digital system, the broadcasters will loose their audience to paid TV. It is technically not feasible to broadcast multiple TV programs or increase media quality within the existing analog channels. It is therefore necessary to adopt a digital transmission system, which provides the solution to both technical issues pointed out.

After years of discussions, the Brazilian consortium decided to base our digital television system, SBTVD-T, in the one made by the Japanese, ISDB-T. The technical aspects that led towards this choice instead of the others were the lower power consumption due to better modulation schemes and the possibility to broadcast to mobile devices as well as fixed devices within the same physical channel.

The purpose of this study arose from the need for a tool to test the set-top box that is being developed in the Signal Processing Laboratory. There was no available tool to multiplex and broadcast the reference elementary streams, necessary to test the reception and decoding of the set-top-box. Hence, the objective of the project was to develop a flexible and free tool that allows for the transmission of multimedia streams according to the Brazilian standard.

To achieve this, it was necessary to study the standards ISO/IEC13818-1

and NBR15603. After some research looking for similar solutions that already existed, FFmpeg was found to be very easily modifiable, and was chosen as starting point of development. As methods to validate the development, a set of four receivers were used, as well as tools to analyse the generated bitstreams.

Although the code of FFmpeg was well commented, the lack of documentation for the tool led to several misuses at first tests. For example, there was no indication of the unit of the option muxrate and it was supposed to be kbps at first. After wrongly generated streams, it was realized that the unit was actually bps. Also, the LATM flag in the multiplexer does not really encapsulate the audio stream in LATM, it only sets the stream type in the PMT stream descriptor. This was discovered after having no audio reception in the EiTV set-top-box receiver while there was audio in the Philips TV receiver. This is probably due to the fact that only the TV has an audio decoder compliant to DTV standards other than SBTVD.

Additionally to the tests that were already performed, extra experimentations can be done by applying reference video frames in the multiplexer input. The current development status allows that. Also, the encoding and multiplexing steps that are done separately could be combined into one, so that the same video input could generate both the HD and 1-Segment services.

Looking back to the objectives defined at the introductory chapter, it can be seen that the proposed tasks were all accomplished. The project delivers a tool which is compliant to the SBTVD standard, transmits video and audio in synchronization and allows multiple services to be broadcasted. The compliance to SBTVD and multiple service support are the two features added by this project. Timing functionalities already existed, although no one in the lab could dedicate enough effort to get it to work before.

The SBTVD compliance was achieved by adding the System Information tables which were not present in the original FFmpeg software. The Network Information Table is being created correctly. This is affirmed because in blind scans the receivers are capable of identifying the channel in their presence. Without it, though, the channel is not identified. The Service Description Table is also correct, as the receivers manage to show the service names when they receive the table.

The analyses carried out with the generated streams show the formation of almost all descriptors, the exceptions are those that the analysers can not decode. All the program and system descriptors added are being correctly interpreted by TS Analyser and TS Parser, as was shown in the tests described in previous chapters.

The SBTVD compliance is not yet fullt implemented, though. Even if it

was proven that the multiplexed stream can be received by the devices and they manage to correctly decode the streams, the Event Information Table, which is mandatory, is not yet implemented. This table is not responsible for any tuning, demodulation of decoding tasks, it is only informational and that is why it was left to further development. The system works without it.

Besides, the Time Offset Table and its mandatory descriptor were implemented as well, but when they were transmitted the time information in the receivers was not updated as indicated in the table all the times. Not much time was dedicated until now to understand why this issue happens, because the multiplexer works without this information. There is no relation between current time display and presentation of the streams in sync.

Most of all, the parts which are not yet implemented are described in the theoretical chapters and can be developed with little effort, using the FFmpeg framework and taking profit of some of the algorithms described by the author.

Future development in this project can be done to turn the interface of the tool more user-friendly. It is possible to create a graphical interface with lists of valid parameter values and the status of the multiplexing process. Additionally, one key feature yet to be implemented is the second transmission profile to allow the creation of transport streams with more than just two services. The profile 2 is already defined and described in the text, and excerpts of code can be reused to implement this feature.

Bibliography

ADVANCED TELEVISION SYSTEMS COMMITTEE. *A/53: ATSC Digital Television Standard*. [S.l.], 2014. Disponível em: <http://www.atsc.org/cms/index.php/standards/standards/50-atsc-a53-standard>. Acesso em: May 10, 2014. Cited in page 27.

ASSOCIATION OF RADIO INDUSTRIES AND BUSINESSES. *GUIDE to ARIB Standards and ARIB Technical Reports*. [S.l.], 2014. Disponível em: <http://www.arib.or.jp/english/html/overview/index.html>. Acesso em: May 10, 2014. Cited in page 28.

ASSOCIAçãO BRASILEIRA DE NORMAS TéCNICAS. *ABNT NBR 15603-2*: Televisão digital terrestre – multiplexação e serviços de informação (si) sintaxes e definições da informação básica de si. Rio de Janeiro, 2007. 135 p. Cited 6 times in pages 50, 51, 52, 53, 113, and 114.

ASSOCIAçãO BRASILEIRA DE NORMAS TéCNICAS. *ABNT NBR 15608-3*: Televisão digital terrestre — guia de operação parte 3: Multiplexação e serviço de informação (si) – guia para implementação da abnt nbr 15603:2007. Rio de Janeiro, 2007. 91 p. Cited in page 51.

BRASIL. *Decreto nº 8.061, de 29 de julho de 2013*. [S.l.], 2010. Disponível em: <http://www.planalto.gov.br/ccivil_03/_Ato2011-2014/2013/Decreto/D8061.htm/>. Acesso em: 25 mar. 2014. Cited 2 times in pages 28 and 29.

CPLUSPLUS.COM. *C Library Reference*. [S.l.], 2014. Disponível em: <http://www.cplusplus.com/reference/clibrary/>. Acesso em: June 08, 2014. Cited in page 59.

DEKTEC. *DTA-115 Multi-Standard VHF/UHF Modulator for PCI Bus Overview*. [S.l.], 2014. Disponível em: <http://www.dektec.com/products/PCI/DTA-115/>. Acesso em: June 08, 2014. Cited in page 71.

DIGITAL VIDEO BROADCASTING PROJECT. *About DVB*. [S.l.], 2014. Disponível em: <https://www.dvb.org/about>. Acesso em: May 10, 2014. Cited in page 27.

EBC, E. B. de C. S. *Famílias do DF já podem marcar consultas pela TV digital*. [S.l.], 2014. Disponível em: <http://www.ebc.com.br/tecnologia/2014/02/familias-do-distrito-federal-ja-podem-marcar-consultas-pela-tv-digital>. Acesso em: May 10, 2014. Cited in page 28.

EITV. *Laboratório ISDB-T DTVi*. [S.l.], 2014. Disponível em: <http://www.eitv.com.br/solucoes/laboratorio-isdb-t-dtvi/>. Acesso em: June 08, 2014. Cited in page 71.

EMBRATEL. *Follheto de Apresentação Stélite Starone C2*. [S.l.], 2014. Disponível em: <http://www.starone.com.br/internas/biblioteca/pdf/folhetoC2.pdf>. Acesso em: 23 mar. 2014. Cited in page 27.

FFMPEG. *FFMPEG About Page*. [S.l.], 2014. Disponível em: <https://ffmpeg.org/about.html>. Acesso em: June 06, 2014. Cited 2 times in pages 25 and 59.

FRAUNHOFER-GESELLSCHAFT. *The AAC audio coding family for broadcast and cable TV*. [S.l.], 2013. Disponível em: <http://www.iis.fraunhofer.de/content/dam/iis/de/dokumente/amm/wp/AAC_Broadcast_CableTV.pdf>. Acesso em: May 19, 2014. Cited 2 times in pages 33 and 34.

FREE SOFTWARE FOUNDATION. *GNU General Public License, version 2*. [S.l.], 2014. Disponível em: <http://www.gnu.org/licenses/old-licenses/gpl-2.0.html>. Acesso em: June 06, 2014. Cited in page 25.

IBGE. *Pesquisa Nacional por Amostra de Domicílios 2011*. [S.l.], 2011. Disponível em: <http://www.ibge.gov.br/home/estatistica/pesquisas/pesquisa_resultados.php?id_pesquisa=40>. Acesso em: 23 mar. 2014. Cited in page 27.

IMMAGINE COMMUNICATIONS. *Selenio Modules*. [S.l.], 2014. Disponível em: <http://www.imaginecommunications.com/products/networking/processing-compression/selenio-media-processing/selenio-modules.aspx>. Acesso em: 25 abr. 2014. Cited in page 55.

INTERNATIONAL STANDARDS ORGANIZATION / INTERNATIONAL ELECTROTECHNICAL COMMISSION. *ISO/IEC 13818-1 International Standard*: Information technology — generic coding of moving pictures and associated audio information: Systems. Geneva, 2013. 174 p. Cited 10 times in pages 36, 37, 38, 39, 40, 41, 67, 105, 111, and 112.

KERNIGHAN, B. W. *The C Programming Language*. 2nd. ed. [S.l.]: Prentice Hall Professional Technical Reference, 1988. ISBN 0131103709. Cited 2 times in pages 59 and 67.

NOW!DIGITAL. *Caixa faz eventos para mostrar uso de interatividade da TV digital*. [S.l.], 2010. Disponível em: <http://idgnow.com.br/mobilidade/2010/02/09/caixa-faz-eventos-para-simular-uso-de-interatividade-da-tv-digital/>. Acesso em: 25 mar. 2014. Cited in page 28.

PUC/RJ. *Ginga About Page*. [S.l.], 2014. Disponível em: <http://www.gingancl.org.br/en/sobre>. Acesso em: May 10, 2014. Cited in page 28.

REDE GLOBO DE TELEVISãO. *Atlas de Cobertura da Rede Globo*. [S.l.], 2014. Disponível em: <http://comercial2.redeglobo.com.br/atlasdecobertura/Paginas/Totalizador.aspx>. Acesso em: June 06, 2014. Cited in page 27.

RICHARDSON, I. E. G. *Video Codec Design*. [S.l.]: Wiley, 2002. Cited 2 times in pages 32 and 33.

UNLIM. *ISDB-T channel, segment and program allocation*. [S.l.], 2014. Disponível em: <http://en.wikipedia.org/wiki/ISDB>. Acesso em: June 06, 2014. Cited in page 95.

VCODEX. *White Paper: An Overview of H.264 Advanced Video Coding*. [S.l.], 2014. Disponível em: <http://www.vcodex.com/h264.html>. Acesso em: May 19, 2014. Cited in page 33.

WATKINSON, J. *The MPEG Handbook*. [S.l.]: Elsevier, 2004. Cited in page 44.

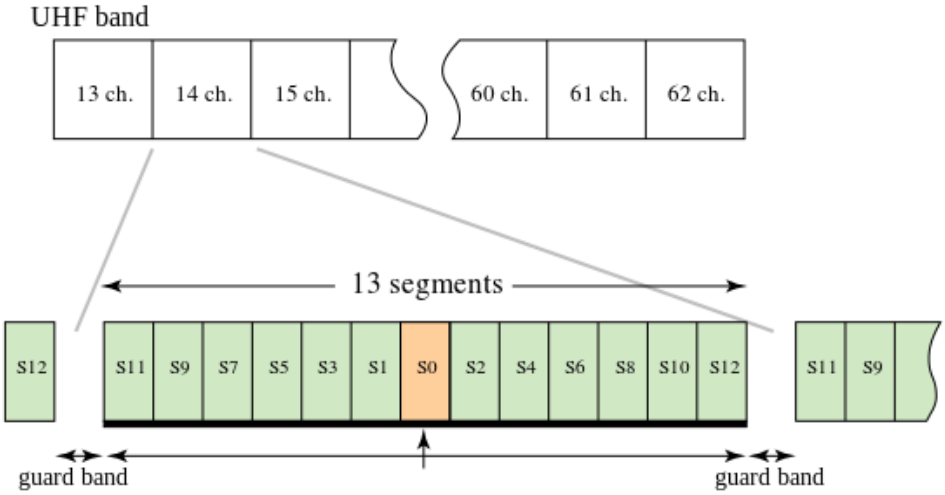
Appendix

APPENDIX A – Modulation aspects of the SBTVD Standard

In ISDB-T, a UHF channel is split into 13 frequency segments. The narrow-band receivers, which are capable of decoding only the central channel frequency(or segment), is commonly called a 1-segment receiver. The wide-band receivers, that can decode all the 13 frequency segments, are called full-segment decoders. Figure 29 shows the segment distribution inside a channel. 'S0' is the central segment for narrow-band tuners. 'S1' to 'S12' are the other segments. As narrower the band, as smaller is the amount of information that can be sent through it. 1-Segment receivers are used in mobile devices, and are capable of receive low definition video only, along with one single audio stream. Full-segment receivers are used in fixed devices, such as Full-HD TVs, and are capable of decode high definition video along with multiple audio streams.

The modulation schemes usually used in 'S0' are different than the ones of the other segments. Since 1-segment TV is targeted for mobile devices, the signal reach should be higher and at most a small dipole antenna is availlable, then QPSK modulation is used. For the other segments, the amount of data is much higher, and customers don't mind to have an amplified antenna attached to the back of their TV, so less robust modulation is applied, such as 64QAM.

Figure 29 – ISDB-T channel, segment and program allocation.



Source: UnLiM, derived from Namazu-tron, obtained in Wikipedia (UNLIM, 2014)

APPENDIX B – Excerpts of code in C language

The following lists present either code developed by the author or parts of the existent code used throughout the text to illustrate the functioning of the tool.

Listing B.1 – Excerpt of MpegTSWrite structure

```
typedef struct MpegTSWrite {
...
MpegTSSection pat;
MpegTSSection sdt;
MpegTSService **services;
...
int nb_services;
...
int mux_rate;
...
int service_id;
int pmt_start_pid;
...
}MpegTSWrite;
```

Listing B.2 – Example of input option

```
{ "mpegts_transport_stream_id", "Set transport_stream_id field
   .", offsetof(MpegTSWrite, transport_stream_id),
   AV_OPT_TYPE_INT, {.i64 = 0x0001 }, 0x0001, 0xffff,
   AV_OPT_FLAG_ENCODING_PARAM}
```

Listing B.3 – Excerpt of MpegTSSection structure

```
typedef struct MpegTSSection {
  int pid;
  int cc;
  void(*write_packet)(struct MpegTSSection *s, const uint8_t *
     packet);
  void *opaque;
} MpegTSSection;
```

Listing B.4 – Excerpt of MpegTSService structure

```
typedef struct MpegTSService {
MpegTSSection pmt;
int sid;
char *name;
char *provider\hspace{0.1mm}\_\hspace{0.1mm}name;
int pcr\hspace{0.1mm}\_\hspace{0.1mm}pid;
...
}MpegTSService;
```

Listing B.5 – Excerpt of MpegTSWriteStream structure

```
typedef struct MpegTSWriteStream {
struct MpegTSService *service;
int pid;
int cc;
...
}MpegTSWriteStream;
```

Listing B.6 – Excerpt of multiple services creation algorithm.

```
    provider_name = provider ? provider->value :
       DEFAULT_PROVIDER_NAME;
  calculated_HD_service_ID = 0x0000;
  calculated_HD_service_ID = ( ts->onid & 0x7FF ) << 5 | 0x0
     << 3 | 0x0;
  service = mpegts_add_service(ts, calculated_HD_service_ID,
     provider_name, "SVC HD Full Seg");
  service->pmt.write_packet = section_write_packet;
  service->pmt.opaque = s;
  service->pmt.cc = 15;

  calculated_LD_service_ID = 0x0000;
  calculated_LD_service_ID = ( ts->onid & 0x7FF ) << 5 | 0x3
     << 3 | 0x1;
  service = mpegts_add_service(ts, calculated_LD_service_ID,
     provider_name, "SVC LD 1-Seg");
  service->pmt.write_packet = section_write_packet;
  service->pmt.opaque = s;
  service->pmt.cc = 15;

  ts->final_nb_services = 2;
```

Listing B.7 – Created macros.

```
#define DEFAULT_NETWORK_NAME    "LaPSI TV - UFRGS"
#define DEFAULT_COUNTRY_CODE    "BRA"
#define DEFAULT_NID   0x0640
#define SDT_RETRANS_TIME 500
#define NIT_RETRANS_TIME 50
#define TOT_RETRANS_TIME 100
```

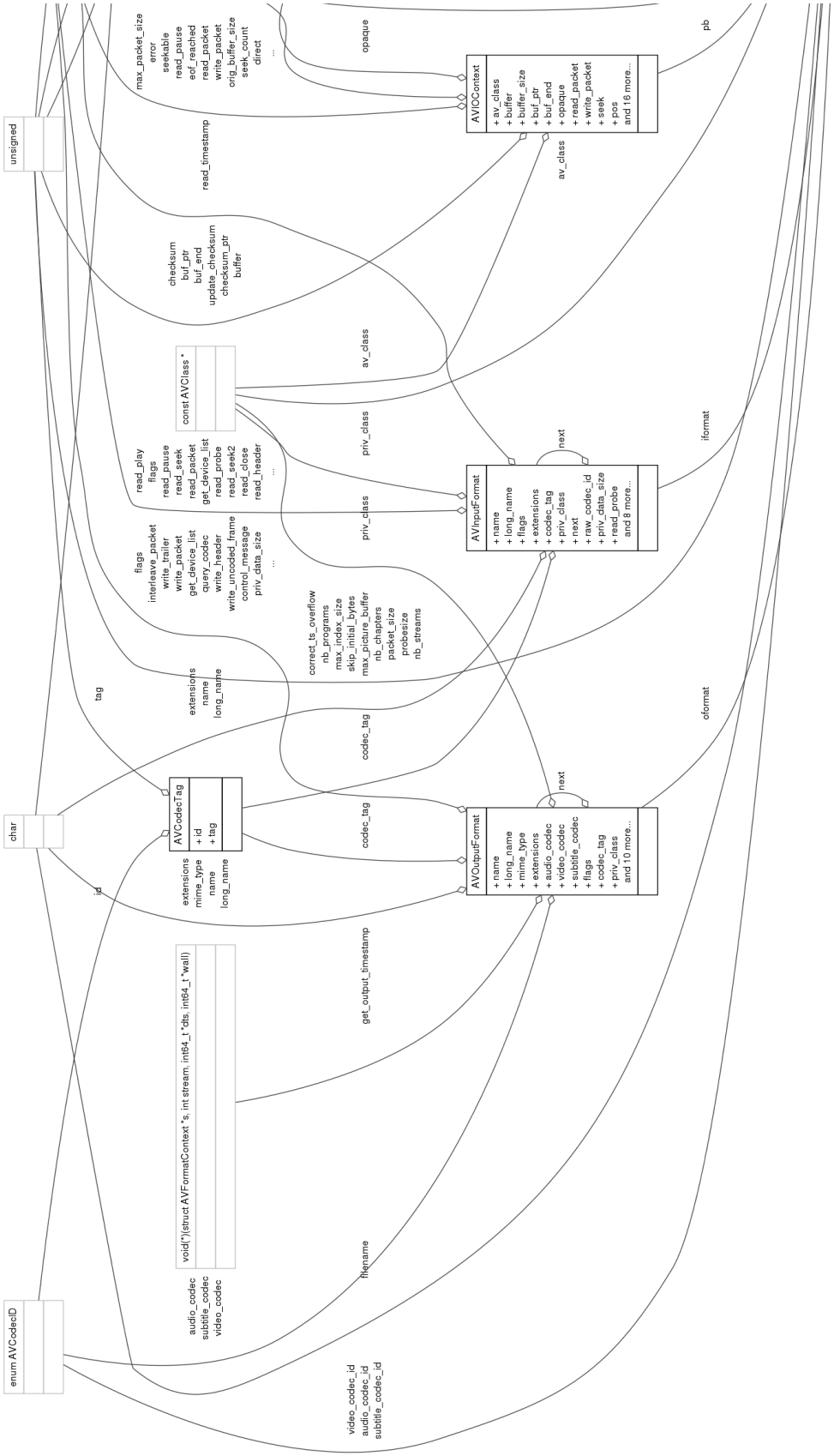
Listing B.8 – Pointer assignment algorithm.

```
q = data;
desc_len_ptr = q;
q += 2;

*q++ = 0x40;
putstr8(&q, DEFAULT_NETWORK_NAME);

*q++ = 0xFE;
sys_mgmt_desc_length_ptr = q;
*q++;
*q++ = 0x03;
*q++ = 0x01;
```

APPENDIX C – Class diagram for AVFormatContext structure

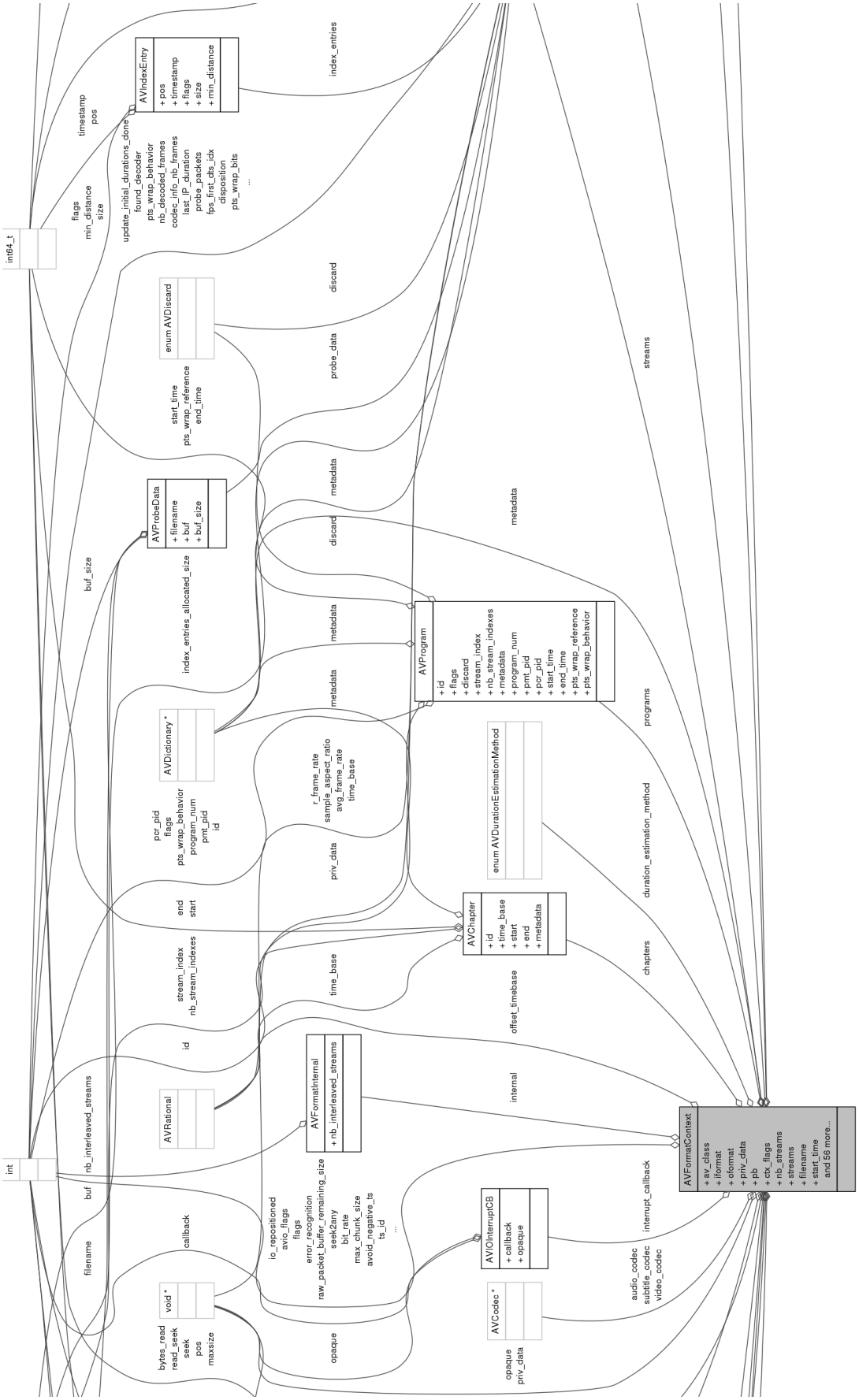
The class diagram for AVFormatContext is split into three parts because it is too big to fit one single page. The parts are supposed to be concatenated side by side, so that the lines representing the relations between classes match.

Figure 30 – Class diagram for AVFormatContext structure, part 1.



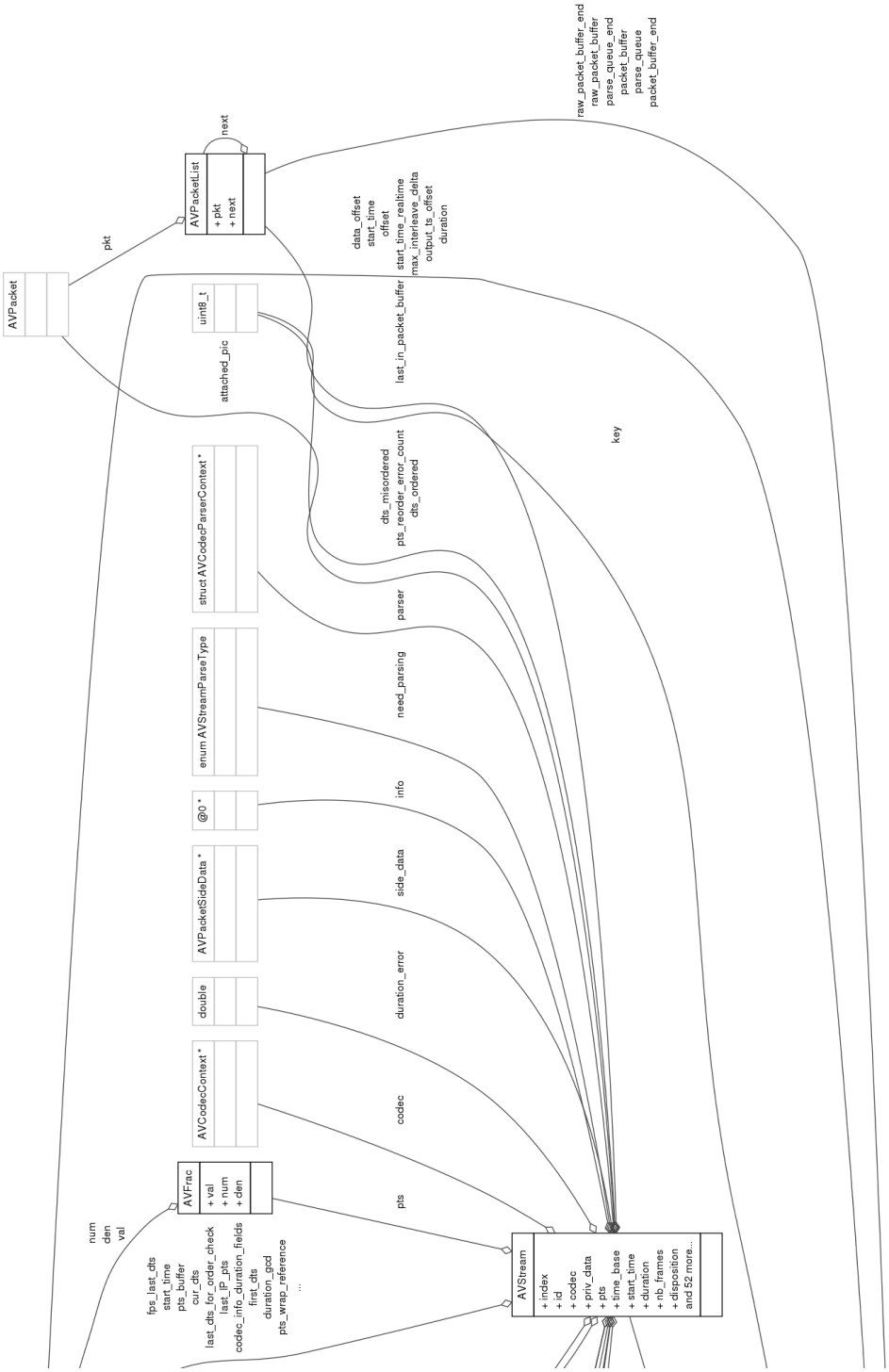
Source: Automatically generated with Doxygen from the source code.

Figure 31 – Class diagram for AVFormatContext structure, part 2.



Source: Automatically generated with Doxygen from the source code.

Figure 32 – Class diagram for AVFormatContext structure, part 3.

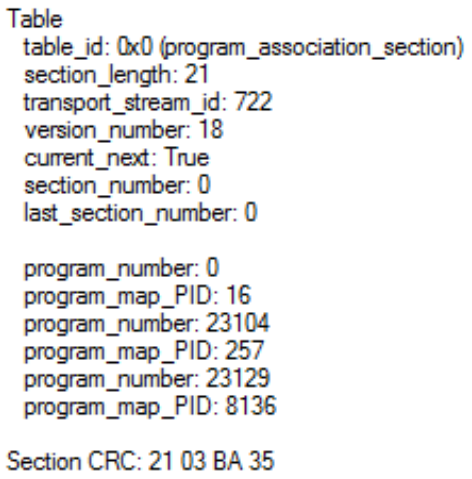


Source: Automatically generated with Doxygen from the source code.

APPENDIX D – Analyses of PSI tables

As the first example, consider Figure 33, that shows the display of a TS Analyser when a PAT table section is analysed. The field *table id* indicates `0x00`, which is the PAT code according to Figure 37. The pair *program_number* `0` and *program_map_PID* `16` is unique and its presence indicates that the NIT table is present and is carried in the PID `16`. It's unique because, according to the ISO standard, a program number `0` always indicates the NIT PID, and the NIT table shall always use the PID `16`, ISO/IEC (2013, 2.4.4.3). The other pairs indicate that there are two programs in the transport stream, with program numbers `23104` and `23129`. Each of these programs have their PMTs allocated in the PIDs `257` and `8136`, respectively. In the subsection 3.5.2, it will be seen how these values are taken into account.

Figure 33 – TS Analyser output showing a PAT Table.



Source: Reproduction of the MPEG TS Analyser software.

Figure 34, that follows, shows a snapshot of the analyser output for a PMT table. In the example, the *program number* is `23104`, the table ID corresponding to the PMT is `0x02` (as defined by ISO13818-1) and the PCR information is carried in the PID (PCR_PID) `256d`. Yet, from Figure 35, it can be seen at the TS Header that the PID carrying the PMT table is number `257`, as indicated by the PAT table analysed before.

By analysing both Table 9 and Figure 34, one may notice that the stream types and PIDs that come after the PCR information match: the ES with PID `273` is H.264 video and the ESs with PIDs `274` and `275` are AAC/LATM audio

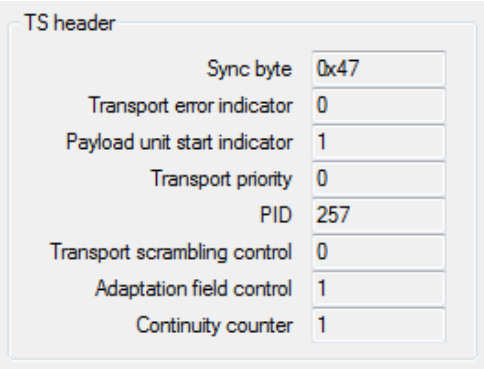
streams, which means that there are two audio streams. The stream with type `6d` and PID `288` is the subtitle / closed captions stream. Later on this chapter the descriptors shown here will be discussed.

Figure 34 – TS Analyser output showing a PMT Table.

```
Table
  table_id: 0x2 (program_map_section)
  section_length: 53
  program_number: 23104
  version_number: 18
  current_next: True
  section_number: 0
  last_section_number: 0

  PCR_PID: 256
  Program_descriptor_length: 0

  stream_type: 27
  elementary_PID: 273
  ES_descriptor_length: 6
  Component tag: 0

  stream_type: 17
  elementary_PID: 274
  ES_descriptor_length: 3
  Component tag: 16

  stream_type: 17
  elementary_PID: 275
  ES_descriptor_length: 3
  Component tag: 17

  stream_type: 6
  elementary_PID: 288
  ES_descriptor_length: 8
  Component tag: 48

Section CRC: 90 75 AB BE
```

Source: Reproduction of the MPEG TS Analyser software.

Figure 35 – TS Analyser output showing the TS Header info for the PMT Table.



Source: Reproduction of the MPEG TS Analyser software.

Annex

ANNEX A – Tables from ISO/IEC13818-1

Figure 36 – PSI tables names, PID numbers and contents description.

Structure Name	Stream Type	PID number	Description
Program Association Table	ITU-T Rec. H.222.0 | ISO/IEC 13818-1	0x00	Associates Program Number and Program Map Table PID
Program Map Table	ITU-T Rec. H.222.0 | ISO/IEC 13818-1	Assignment indicated in the PAT	Specifies PID values for components of one or more programs
Network Information Table	Private	Assignment indicated in the PAT	Physical network parameters such as FDM frequencies, Transponder Numbers, etc.
Conditional Access Table	ITU-T Rec. H.222.0 | ISO/IEC 13818-1	0x01	Associates one or more (private) EMM streams each with a unique PID value
Transport Stream Description Table	ITU-T Rec. H.222.0 | ISO/IEC 13818-1	0x02	Associates one or more descriptors from Table 2-39 to an entire Transport Stream

Source and Copyright: ISO/IEC (2013, 2.4.4)

Figure 37 – PSI tables IDs.

Value	description
0x00	program_association_section
0x01	conditional_access_section (CA_section)
0x02	TS_program_map_section
0x03	TS_description_section
0x04	ISO_IEC_14496_scene_description_section
0x05	ISO_IEC_14496_object_descriptor_section
0x06-0x37	ITU-T Rec. H.222.0 | ISO/IEC 13818-1 reserved
0x38-0x3F	Defined in ISO/IEC 13818-6
0x40-0xFE	User private
0xFF	forbidden

Source and Copyright: ISO/IEC (2013, 2.4.4.4)

Table 9 – Some Standardized Elementary Stream Types.

Decimal Value	Hex Value	Description
6	0x06	ITU-T Rec. H.222 and ISO/IEC 13818-1 (MPEG-2 packetized data) privately defined (ie, DVB subtitles/VBI and AC-3)
17	0x11	ISO/IEC 14496-3 (MPEG-4 LOAS multi-format framed audio) in a packetized stream
27	0x1B	ITU-T Rec. H.264 and ISO/IEC 14496-10 (lower bit-rate video) in a packetized stream

Source: (ISO/IEC, 2013, 2.4.4.9)

ANNEX B – Tables from ABNT NBR15603

Figure 38 – PSI tables PIDs according to SBTVD standard.

Tabela	PID
PAT [a]	0x0000
PMT [a]	Designado indiretamente pela PAT
CAT [a]	0x0001
NIT [a]	0x0010
SDT	0x0011
BAT	0x0011
EIT	0x0012
EIT (transmissão de televisão digital terrestre)	0x0012, 0x0026, 0x0027
RST	0x0013
TDT	0x0014
TOT	0x0014
PCAT	0x0022
BIT	0x0024
NBIT	0x0025
LDT	0x0025
ST	Exceção 0x0000, 0x0001, 0x0014
Pacotes nulos [a]	0x1FFF
[a] Conforme a ARIB STD-B10.	

Source and Copyright: ABNT (2007a, Part 2 - 7.1.4)

Figure 39 – PSI tables IDs according to SBTVD standard.

Table_id	Tabela	Nível de transmissão	Ciclo de transmissão
0x00	PAT	Obrigatório	Um ou mais/100 ms
0x01	CAT	Obrigatório	Um ou mais/1 s
0x02	PMT	Obrigatório	Um ou mais/100 ms
0x40	NIT (rede atual)	Obrigatório	Um ou mais/10 s
0x41	NIT (outra rede)	Opcional	Um ou mais/10 s
0x42	SDT (feixe atual)	Obrigatório	Um ou mais/2 s
0x46	SDT (outro feixe)	Opcional	Um ou mais/10 s
0x4A	BAT	Opcional	Um ou mais/10 s
0x4E	EIT (programa presente e futuro do feixe atual)	Obrigatório	Um ou mais/2 s
0x4F	EIT (programa presente e futuro de outro feixe)	Opcional	Um ou mais/10 s
0x50 – 0x5F	EIT (programa em até 8 dias do feixe atual)	Opcional	Um ou mais/10 s
0x50 – 0x5F	EIT (programa após 8 dias do feixe atual)	Opcional	Um ou mais/30 s
0x60 – 0x6F	EIT (programa em até 8 dias de outro feixe)	Opcional	Um ou mais/10 s
0x60 – 0x6F	EIT (programa após 8 dias de outro feixe)	Opcional	Um ou mais/30 s
0x70	TDT	Opcional	Um ou mais/30 s
0x71	RST	Opcional	Opcional
0x72	ST	Opcional	Opcional
0x73	TOT	Obrigatório	Um ou mais/30 s

Source and Copyright: ABNT (2007a, Part 2 - 7.1.4)