



INTERNATIONAL TELECOMMUNICATION UNION

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

P.1201.2

Amendment 1

(05/2013)

SERIES P: TERMINALS AND SUBJECTIVE AND
OBJECTIVE ASSESSMENT METHODS

Models and tools for quality assessment of streamed
media

Parametric non-intrusive assessment of audiovisual
media streaming quality - higher resolution
application area

Amendment 1

CAUTION !

PREPUBLISHED RECOMMENDATION

This prepublication is an unedited version of a recently approved Recommendation. It will be replaced by the published version after editing. Therefore, there will be differences between this prepublication and the published version.

FOREWORD

The International Telecommunication Union (ITU) is the United Nations specialized agency in the field of telecommunications, information and communication technologies (ICTs). The ITU Telecommunication Standardization Sector (ITU-T) is a permanent organ of ITU. ITU-T is responsible for studying technical, operating and tariff questions and issuing Recommendations on them with a view to standardizing telecommunications on a worldwide basis.

The World Telecommunication Standardization Assembly (WTSA), which meets every four years, establishes the topics for study by the ITU-T study groups which, in turn, produce Recommendations on these topics.

The approval of ITU-T Recommendations is covered by the procedure laid down in WTSA Resolution 1.

In some areas of information technology which fall within ITU-T's purview, the necessary standards are prepared on a collaborative basis with ISO and IEC.

NOTE

In this Recommendation, the expression "Administration" is used for conciseness to indicate both a telecommunication administration and a recognized operating agency.

Compliance with this Recommendation is voluntary. However, the Recommendation may contain certain mandatory provisions (to ensure, e.g., interoperability or applicability) and compliance with the Recommendation is achieved when all of these mandatory provisions are met. The words "shall" or some other obligatory language such as "must" and the negative equivalents are used to express requirements. The use of such words does not suggest that compliance with the Recommendation is required of any party.

INTELLECTUAL PROPERTY RIGHTS

ITU draws attention to the possibility that the practice or implementation of this Recommendation may involve the use of a claimed Intellectual Property Right. ITU takes no position concerning the evidence, validity or applicability of claimed Intellectual Property Rights, whether asserted by ITU members or others outside of the Recommendation development process.

As of the date of approval of this Recommendation, ITU [had/had not] received notice of intellectual property, protected by patents, which may be required to implement this Recommendation. However, implementers are cautioned that this may not represent the latest information and are therefore strongly urged to consult the TSB patent database at <http://www.itu.int/ITU-T/ipr/>.

© ITU 2013

All rights reserved. No part of this publication may be reproduced, by any means whatsoever, without the prior written permission of ITU.

Amendment 1 to Recommendation ITU-T P.1201.2

Parametric non-intrusive assessment of audiovisual media streaming quality - higher resolution application area

Summary

Recommendation ITU-T P.1201.2 specifies the algorithmic model for the higher resolution (HR) application area of ITU-T P.1201.

The ITU-T P.1201 series of Recommendations specifies models for monitoring the audio, video and audiovisual quality of IP-based video services based on packet-header information. The higher resolution application area of the ITU-T P.1201.2 part of ITU-T P.1201 can be applied to the monitoring of performance and quality of experience (QoE) of video services such as IPTV. The algorithm for the lower resolution case is specified in ITU-T P.1201.1. See ITU-T P.1201 for details and respective application ranges.

Amendment 1 to Recommendation ITU-T P.1201.2

Parametric non-intrusive assessment of audiovisual media streaming quality - higher resolution application area

1. Scope

Recommendation ITU-T P.1201.2 specifies the algorithmic model for the higher resolution (HR) application area of ITU-T P.1201. ITU-T P.1201 specifies models for monitoring the audio, video and audiovisual quality of IP-based video services based on packet-header information. The higher resolution application area of the ITU-T P.1201.2 part of ITU-T P.1201 can be applied to the monitoring of performance and quality of experience (QoE) of video services such as IPTV. The algorithm for the lower resolution case is specified in ITU-T P.1201.1. See ITU-T P.1201 for details and respective application ranges.

2. References

The following ITU-T Recommendations and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published.

The reference to a document within this Recommendation does not give it, as a stand-alone document, the status of a Recommendation.

[ITU-T P.1201.1]	Recommendation ITU-T P.1201.1 (2012), <i>Parametric non-intrusive assessment of audiovisual media streaming quality – lower resolution application area</i> .
[ITU-T H.264]	Recommendation ITU-T H.264, <i>Advanced video coding for generic audiovisual services</i> .
[ITU-T H.222.0]	Rec. ITU-T H.222.0 ISO/IEC 13818-1, <i>Information technology -- Generic coding of moving pictures and associated audio information: Systems</i> .
[IETF RFC 2250]	IETF RFC 2250 (1998), <i>RTP Payload Format for MPEG1/MPEG2 Video</i> .
[IETF RFC 3550]	IETF RFC 3550 (2003), <i>RTP: A Transport Protocol for Real-Time applications</i> .
[IETF RFC 3640]	IETF RFC 3640 (2003), <i>RTP Payload Format for Transport of MPEG-4 Elementary Streams</i> .
[IETF RFC 4184]	IETF RFC 4184 (2005), <i>RTP Payload Format for AC-3 Audio</i> .
[IETF RFC 4566]	IETF RFC 4566 (2006), <i>SDP: Session Description Protocol</i> .
[IETF RFC 6184]	IETF RFC 6184 (2011), <i>RTP Payload Format for H.264 Video</i> .
[ISO/IEC 13818-3]	ISO/IEC 13818-3:1998, <i>Information technology -- Generic coding of moving pictures and associated audio information -- Part 3: Audio</i> .

- [ISO/IEC 14496-10] ISO/IEC 14496-10:2012, *Information technology -- Coding of audio-visual objects -- Part 10: Advanced Video Coding*.
- [ISO/IEC 14496-3] ISO/IEC 14496-3:2009, *Information technology -- Coding of audio-visual objects -- Part 3: Audio*

3. Abbreviations and acronyms

This Recommendation uses the following abbreviations and acronyms:

CTS	Continuous Timestamp
DTS	Decoding Timestamp
ES	Elementary Stream
GOP	Group of Pictures
HD	High Definition (television)
HR	Higher Resolution
HBR	High Bit Rate
IP	Internet Protocol
IPTV	Internet Protocol Television
ITU	International Telecommunications Union
ITU-T	ITU Telecommunications Standardization Sector
MOS	Mean Opinion Score
MPEG	Motion Pictures Expert Group
NTSC	National Television Standard Committee (60-Hz TV, used mainly in US and Canada)
NCTS	Non-continuous Timestamp
PAL	Phase Alternating Line (50-Hz TV, used in Europe and elsewhere)
PAT	Program Association Table
PCR	Program Clock Reference
PES	Packetized Elementary Stream
PMT	Program Map Table
PTS	Presentation Timestamp
RTP	Real Time Protocol
TS	Transport Stream
UDP	User Datagram Protocol

4. ITU-T P.1201.2 (HR) algorithm description

4.0. Introduction

The following description provides text, formulae and pseudocode to explain the ITU-T P.1201.2 algorithm.

The algorithm estimates three Mean Opinion Score (MOS) values for audio quality, video quality, and audiovisual quality, considering a measurement window of a defined length of 10 seconds.

The algorithm derives the MOS by analysing the bitstream in four stages. This analysis is carried out one step after the other covering the entire 10 second interval. Each stage can be seen as a complete sub-process in the sense that it receives input values and generates output values. The stages are defined as follows.

- **Packet and Global Info Acquisition:** At stage 1, the data for a predefined time window of audio and video is collected, and the relevant basic data is extracted. Header-based

information is pre-shaped for the use in subsequent steps. In addition, side information is gathered from the available sources (see ITU-T P.1201 and detailed text below) or from SDP (session description protocol).

- **Frame-based Parameter Extraction:** At stage 2, the sequence of audio and video frames is extracted from the packet headers. The beginning of the data extraction for video is aligned with the appearance of an I-frame. For audio, capturing starts with the appearance of the first packetized elementary stream structure after that I-frame. Quality-relevant features, such as frame types, bitrate and packet loss information are derived from the extracted information. In case that the content is encrypted or the sampled sequence contains packet loss, it may not be possible to derive all data elements exactly, and hence the values for missing data elements are estimated. Since the measurement window needs to comprise an integer number of GOPs, capturing stops at the end of the GOP after the 10 second measurement window has been completed.
- **Sequence-based Parameter Extraction:** At stage 3, parameters are extracted that relate to the sequence of frames comprising the measurement interval. The frame types such as I, P, B frames are detected, and the GOP structure is estimated. Depending on the level of encryption and whether the RTP timestamp is continuous or not, more of the required information has to be estimated, since it cannot be read directly from the bitstream. Also, the number of lost video frames is determined and the lost packets are allocated to the corresponding video frames. Similarly, the lost audio packets are allocated to the corresponding audio PES frames.
- **MOS Estimation:** Stage 4 finally combines the extracted quality-relevant features and provides the estimated MOS values.

Note: The bitstream may either have a pure RTP format, where the audio and video parts appear in separate RTP bitstreams, or audio and video can be multiplexed in an MPEG-2 Transport Stream. The MPEG-2 Transport Stream then may be packetized into an RTP bitstream, or it may be encapsulated in MPEG-2 TS/UDP (i.e., without RTP headers). The quality assessment algorithms and models described in this Recommendation have only been evaluated for MPEG-2 TS/RTP/UDP bitstreams.

In addition, the bitstreams may be encrypted or unencrypted. The MPEG-2 Transport Streams may be encrypted either at the level of packetized elementary streams (PES-Encr), or at the level of the Transport Stream (TS-Encr). In addition, the RTP timestamp is handled in varying forms. In case of TS over RTP, the RTP timestamp may be non-continuous or continuous (below called NCTS or CTS). If it is non-continuous, the RTP timestamp only changes with the video frame. If the RTP timestamp changes continuously, then the timestamp increases proportionally to the elapsed time, in every new RTP packet. These different versions/cases, as well as the case without no -RTP, at all need to be considered during the extraction of parameters. In case of packet loss and even more in combination with encryption, some of the needed information cannot be derived in an exact manner and needs to be estimated, as describe in detail below.

Figure 1 illustrates the overall algorithm of the ITU-T P.1201.2 model in form of a block diagram. The following sections describe the processing in the four stages of this algorithm in more detail.

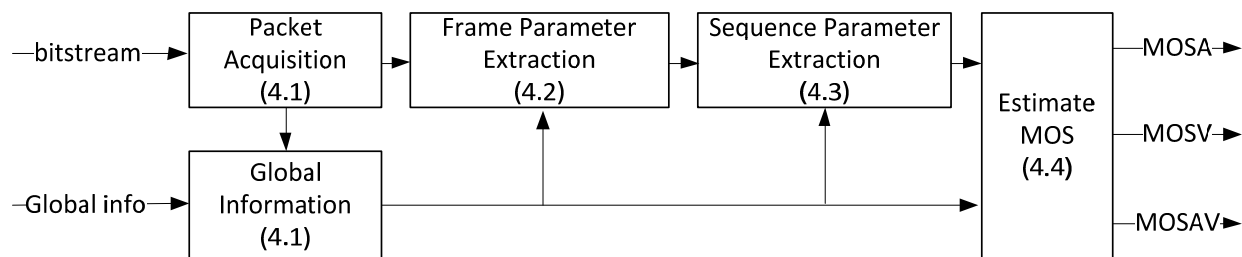


Figure 1 - Block diagram of the ITU-T P.1201.2 model

4.1. Packet Acquisition and Side Information

The packet acquisition stage receives as input a bitstream of IP packets, either read from a network interface or read from a file (e.g. network traces in the form of PCAP files). The relevant information is extracted from the packets, with the packets being collected for one measurement window. Due to network degradations (e.g. packet loss, delay, or jitter), the received bitstream may be affected by packet losses or packet drops. Further, the payload of the packets might be encrypted, so that certain payload-related bitstream or header information is not available. In addition, some information may be available from other sources (e.g. transmitted out-of-band). This type of information is referred to as side information below.

For each stage of processing, this document describes the respective input and output information, as well as the processing component itself.

Input

- Side information file
 - o Destination port
 - o Audio frame duration or audio sample rate
 - o Concealment type
 - o Number of slices
- SDP
- Bitstream of IP packets

Output

```

{ "GlobalInfo": {
  "Video": {
    "Codec"           : <string>,      // H264/AVC
    "Resolution"      : <string>,      // SD, HD720, HD1080
    "FrameRate"       : <number>,
    "ConcealmentType" : <number>,      // 0=Slicing-Mode, 1=Freezing-Mode
    "NumSlices"       : <number>,      // Number of slices per frame
    "isInterlaced"    : <number>      // 0:progressive, 1: interlaced
  },
  "Audio": {
    "Codec"           : <string>,      // AAC-LC, AC3, Mp2, HeAAC
    "SampleRate"      : <number>,
    "FrameLength"     : <number>,      // in Samples
    "FrameDuration"   : <number>,      // audio frame duration (in sec.)
    "Bitrate"         : <number>
  }
}

```

```

}

{ "StrippedUDPPackets": [
    { "PacketArrivalTime" : <number>,
      "PayloadSize"       : <number>,
      "SequenceNr"        : <number>,    // not available in UDP TS case
      "RTPTimeStamp"      : <number>,    // not available in UDP TS case
      "StrippedTSPackets": [
          {
              "MediaIndex" : <number>,    // 0: video, 1: audio
              "CC"          : <number>,    // TS continuity counter
              "PayloadStart" : <number>,    // TS payload unit start indicator
              "PayloadLength" : <number>,    // TS payload length
              "RandomAccess" : <number>,    // Random access indicator
              "PTS"          : <number>,    // media index 0|1, if available
              "DTS"          : <number>,    // media index 0, if available
              "ESFlagsVideo" : <number>,    // resolution, framerate
              "ESFlagsAudio" : <number>,    // sample rate, number of audio channels
              "AssumedFrameEnd": <boolean> // media index 0|1
          } ,
          {
              // next TS Packet; usually in total 7 TS packets
          }
      ]
    } ,
    {
        // next UDP packet
    }
]
}

```

Process

For processing the UDP or RTP bitstreams containing the MPEG-2 Transport Stream the following IETF RFC documents and the MPEG-2 Systems standard have to be considered: RFC 3550 (RTP), RFC 2250, and Rec. H.222. Furthermore, H.264/AVC, MPEG-1-LII, AAC and AC3 specifications are needed to separate the related elementary streams. Note that from the latter only knowledge about the audio frame structure is required for this standard.

The result of the first processing stage is the global info data structure `GlobalInfo` providing the global parameters of the audio and video codec and system parameter under consideration and a list of "stripped" packets providing only the parameters that are relevant for further processing. The global parameters cannot always be derived completely from the bitstream when the bitstream is encrypted. Hence, in a first step, the parameters are initialized with the most likely values. In a second step, further external sources with system configuration data, e.g. provided in a file, are considered. These parameters overwrite the initialized parameters. If during the further processing of the input bitstream the parameters can be derived in a more exact manner, the previous values are overwritten again.

The extraction of the essential information from the Transport Stream relies on scanning the contained elementary streams from the TS PAT/PMT data fields (as defined in Rec. H.222). If more than one audio or video elementary stream is contained, it is assumed that the selection of the appropriate audio and video is provided e.g. by using additional side information.

Most of the values for `GlobalInfo` are derived directly from side information sources or the packet bitstream. Only the following four parameters, namely `Video.Resolution`, `Video.FrameRate`,

`StrippedTSPackets.AssumedFrameEnd` and `Audio.FrameDuration` are obtained by interrelation of parameters or specific transmission properties, as detailed below.

Initialization:

Before processing the side information, the following parameters of the `GlobalInfo` are initialized to the selected values to ensure the normal execution of the algorithms even if some side information variables are missing or are not known. These initialized values are overwritten if they are available from the side information or the bitstream.

```
{ "GlobalInfo": {
  "Video": {
    "FrameRate"       : 50,
    "ConcealmentType" : 1,
    "Resolution"      : "HD720",
    "NumSlices"       : 1,
    "isInterlaced"    : 0
  },
  "Audio": {
    "SampleRate"      : 48000
  }
}
```

Framerate:

The `Video.FrameRate` can be derived from the DTS timestamps in case of PES encryption, or it can be derived from the RTP timestamp for the non-continuous timestamp case (e.g. as the inverse of the timestamp difference between frames). Only in the case of continuous RTP timestamp with encrypted TS payload, the frame rate cannot be derived from the bitstream. In this case, the video frame rate must be provided by system configuration information. Note that the `Video.FrameRate` value can also be parsed from the video elementary stream headers directly, if the bitstream is not encrypted.

Resolution:

Typically, the `Video.Resolution` is parsed from system configuration information. Note, that the `Video.Resolution` value can also be parsed from the video elementary stream headers directly if the bitstream is not encrypted.

CheckFrameEnd:

Input

Bitstream of IP packets

Output

`StrippedUDPPackets.StrippedTSPackets.AssumedFrameEnd`

Process

Since an ES frame always starts at the beginning of a TS packet, the end of the previous ES-frame is in general shorter than the full payload portion of the TS packet. Consequently, the last packet of an ES frame needs some stuffing to fill the rest of that TS packet. Stuffing is always done by creating an adaptation field of the appropriate length. In this algorithm it is assumed that any stuffing that occurs in the adaptation-field is due to a ES frame end. If stuffing is detected, the

variable. AssumedFrameEnd is set for this ES frame to mark that this ES frame has ended regularly.

AudioFrameDuration:

The audio sample rate can typically be derived from the side information. The audio frame duration can be calculated by dividing the audio frame length (of the specific audio codec) by the sample rate. In rare cases, the audio frame duration may be provided as side information directly.

Pseudocode

```
{ "AudioFrameLength": {
    "Mp2"      : 1152,    // samples
    "AC3"      : 1536,
    "AacLC"    : 1024,
    "HeAac"    : 2048
  }
}

using GlobalInfo.Audio {
    FrameLength  = AudioFrameLength[Codec] ;
    FrameDuration = FrameLength / SampleRate ;
}
```

4.2. Frame-based Parameter Extraction

Input

StrippedUDPPackets and GlobalInfo

Output

```
{ "VideoFrames": [
    { "FrameSize"      : <number>,
      "Num_IP"         : <number>,
      "Num_TS"         : <number>,
      "Num_LostTS"     : <number>,
      "Num_LostIP"     : <number>,
      "LossEvents"     : <number>,
      "IsFrozen"       : <boolean>,
      "FrameStartLost" : <boolean>,
      "FrameEndLost"   : <boolean>,
      "LossPosition_TS": <number>,
      "LossPosition_IP": <number>,
      "LossSize"       : <number>,
      "DTS_Incr"       : <number>,
      "PTS_Incr"       : <number> },
    {
        // next video frame
    }
]
}

{ "AudioPESFrames": [
    {
        "Num_IP"      : <number>,
        "Num_TS"      : <number>,
        "FrameSize"   : <number>,
        "Num_LostTS"   : <number>,
        "Num_LostIP"   : <number>,
        "LossEvents"   : <number>,
        "Num_LostAF"   : <number>,
        "PTS_Incr"     : <number>
    },
    ...
]
```

```

    {
        // next audio PES frame
    }
}
}

```

Process

The second processing stage converts the bitstream of packets into arrays of audio and video frames by performing the following tasks:

- Re-ordering of RTP packets that do not appear in ascending order based on the Sequence Number.
- Detecting packet losses by exploiting `StrippedUDPPackets.SequenceNr`, which is a reliable measure. In the MPEG2-TS/UDP case, where `StrippedUDPPackets.SequenceNr` is not available `StrippedUDPPackets.StrippedTSPackets.CC` is used instead, which can fail for a huge number of lost packets.
- Detecting frame boundaries, by observing the TS payload_unit_start_indicator.
- Detecting lost video-frame boundaries, by observing RTP timestamp changes if valid ~~(in~~ (which is not the case of for RTP packets with non-continuous timestamps or in the for MPEG2-TS/UDP case) or alternatively by observing changes in the video DTS (which has a lower accuracy). If the video DTS is not available either, due to TS-level encryption, statistical measures are used to detect the number and position of lost video PES headers, as detailed in the description of the third processing stage, when the data for all frames contained in the measurement window are available.
- Detecting lost audio-frame boundaries. Either the audio PTS or the payload_unit_start_indicator is used. If the PTS is available the PES frame duration and consequently the loss of PES frame headers can be derived from it. If the PTS is not available due to TS-level encryption then the payload_unit_start_indicator is used in combination with a PES frame duration derived from the number of PES frames are observed in the measurement interval.
- Measuring the size of each video and audio frame by adding up the payload size of all packets belonging to this frame.
- Noting the number of lost TS packets, lost UDP-packets and the number of loss-events for each frame in the measurement window.
- Noting the relative position of each loss and the loss-length in a frame.

Figure 2 shows a block diagram illustrating the procedure for processing the received packets to extract the frame-related parameters `VideoFrames` and `AudioFrames`. The functions of this diagram and the modules “Process video TS packet” and “Process audio TS packet” are separately presented below.

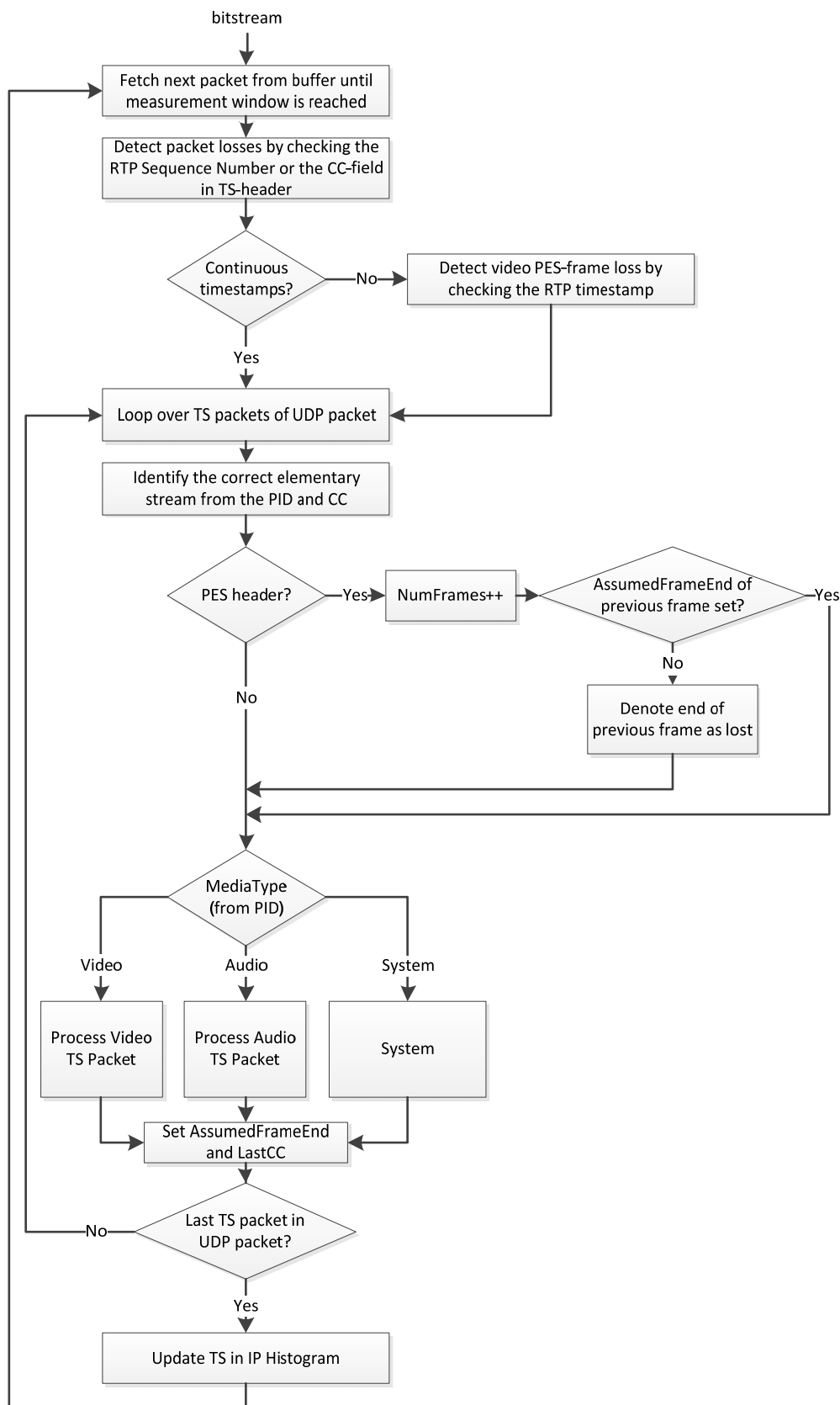


Figure 2 - Block diagram of the second processing stage with the extraction of frame-based information by processing the received packets in the buffer

4.2.1. Fetch next packet from buffer:

Input

StrippedUDPPackets[]

Output

StrippedUDPPackets[]

Process

At this stage, it is assumed that the packets of the input bitstream have been re-ordered in the buffer of stage 1 according to the RTP sequence number, if necessary, and can now be read sequentially, until the end of the measurement window is reached.

4.2.2. Detect packet losses for MPEG2-TS/RTP/UDP packets

Input

StrippedUDPPackets[].RTPSequenceNr

Output

{ "LostRTPPackets": <number> }

Process

In this step, the sequence number of the current packet is compared with the sequence number of the previous packet. The difference should always be 1. Every difference larger than one is considered a packet loss, where a difference of $N > 1$ denotes a loss of $N-1$ packets. Note that the SequenceNumber field in the RTP header has 16 bit and therefore wraps around from 65535 to 0.

4.2.3. Detect packet losses for MPEG2-TS/UDP packets

If MPEG2-TS/UDP based packets are evaluated, no RTP-header will be available. In that case the RTP sequence number is not available and packet losses can only be detected with the continuity counter field StrippedUDPPackets.StrippedTSPackets.CC of the TS-header.

This field works exactly as the RTP-sequence number, but unfortunately only has 4 bits, which means that only the loss of 1-15 TS-packets per media-stream can be detected reliably. For loss detection only the difference between the CC's of the first TS packet of a particular stream in the current packet and the last CC of the same stream in the previous packet are considered.

-Since one UDP-packet may contain up to 7 TS-packets of a single media-stream, a maximum of 2 lost UDP-packets can be detected reliably (resulting in a CC difference of up to 15). For instance a CC difference of 2 would normally mean "one TS is lost" it could also imply the loss of 17 TS-packets of a specific media-stream though.

In a regular scenario, where the TS contains video and audio, the relative frequency of occurrence of audio and video packets can be used to facilitate more precise results of the number of losses. In general the frequency of video TS packets is considerably higher than the one for audio TS packets. Let the ratio of frequency for video and audio packets be:

$$VA \text{ Ratio} = \text{Average TSperPacket(VIDEO)} / \text{Average TSperPacket(AUDIO)}$$

With VA Ratio larger than 1.0, a certain detected loss of audio TS-packets implies a higher loss of video-packets, which should be close to:

$$\text{AssumedLossOfVideoTS} \sim \text{DetectedLossOfAudioTS} * \text{VA_Ratio}$$

Since audio TS packets are not that frequent and are distributed among video- and system packets, ~~it hardly ever happens that~~ rarely more than 15 audio TS packets are lost in one loss event. Therefore, the number of TS-losses for the audio stream can be assumed to be reliable. With a roughly constant frequency ratio *VA Ratio*, the number *AssumedLossOfVideoTS* should be a reliable estimation of the true video loss as well.

The precise number of lost video TS² packets can be found, by taking into account that the CC-difference

$$\text{CC_Diff}(V) = \text{CC}_v[i] - \text{CC}_v[i-1] \quad i = \text{index of current packet}$$

is exact, but only contains the 4 least significant binary digits. Therefore the true number of lost video TS packets is $16*n + \text{CC_Diff}(V)$ where n is the yet unknown number of “wrap arounds” of *CC Diff(V)*. The best estimation for the true number of video TS losses can be found by:

$$n_{opt} = \min (\text{abs}(16*n + \text{CC_Diff}(V) - \text{AssumedLossOfVideoTS}) \text{ for } n=0\dots N$$

$$\text{EstimatedLossOfVideoTS} = 16* n_{opt} + \text{CC_Diff}(V)$$

As mentioned above, for audio

$$\text{EstimatedLossOfAudioTS} = \text{CC_Diff}(A)$$

is already a reliable assumption. This estimation rarely fails. It requires the presence of audio, though.

In the case where no audio is present, enhancements for the estimation-precision of the number of lost TS packets ~~will~~ can be achieved at the sequence-based parameter extraction stage, i.e. when the frame types and average frame sizes per frame type have become known.

4.2.3.4.2.4. Detect video PES-frame loss:

Input

StrippedUDPPackets[].RTPTimestamp, LostRTTPackets

Output

```
{ "PES_Missing": <number> } ,
{ "FrameTime" : <number> }
```

Process

In a second step, the RTP timestamp is examined. The RTP timestamp is only exploitable in this algorithm, if it represents a so called “non-continuous timestamp (NCTS)”. NCTS means that the RTP timestamp is synchronized with the video DTS. It is therefore constant for all RTP packets carrying one video-frame and then abruptly changes for subsequent packets carrying data of the next frame.

In contrary, if the RTP Timestamp is in “continuous timestamp” mode (CTS) mode, its value is increased in equal increments for every packet. The amount of change may be equal for one frame, or group of frames up to the whole sequence. The increment is much smaller than in the NCTS-case. The RTP timestamp is then directly related to the PCR (program clock reference) of the stream.

Similar to the DTS in the TS-header, ~~For the RTP streams the~~ timestamp is based on a 90000 Hz reference. For example, this means for NCTS and a 50 Hz video, a timestamp increment of 1800 per frame. The CTS/NCTS detection is done by observing the frequency and amount of RTP timestamp changes. If for 4 subsequent packets, every packet had a timestamp change, but the change per packet was lower than 0.0133 sec (75 Hz), CTS is assumed.

~~For In~~ the NCTS case, the ~~NCTS RTP timestamp in the RTP Header~~ is set to the decoding time of the first video TS packet of the currently transmitted video PES frame. Consequently, the RTP timestamp changes in the next TS packet which contains SRBI ~~after the packet that contained the TS packet with the next PES header~~, except when the PES header is in the first TS packet of the RTP packet (see note 1 in the pseudocode). If one (or more) PES headers are in a packet immediately before a loss, this will cause a jump in the timestamp, which still may not signal a PES header loss (see note 2 in pseudocode).

Pseudocode

```
set n to the current packet index;
CurrTimeDiff = (n == 0)? 0 : (StrippedUDPPackets[n].TimeStamp - StrippedUDPPackets[n-1].TimeStamp);
PES_Losses = 0;
if( (CurrTimeDiff > 0) && (StrippedUDPPackets[n-1].TimeStamp != -1) )
{
    if( !LostRTPPackets )                // No Loss?
        FrameTime = CurrTimeDiff ;      // CurrTimeDiff is regular time difference between 2 frames
    else
    {
        if( FrameTime > 0 )                // if initialized already
            PES_Losses = CurrTimeDiff / FrameTime ;
        else
            PES_Losses = CurrTimeDiff/3600 ; // if not initiated: assume 25Hz -> FrameTime: 90000 / 50
        if( IsTS )                        // do we have a transportstream or just RTP?
        {
            if( (StrippedUDPPackets[n] has TS packet with VideoPES) &&
                (VideoPES is in first videoTS of StrippedUDPPackets[n]) )
                PES_Losses-- ;           // Note (1) in text above
            if( RTP-Packet immediately before the loss had PES headers ) // Note (2) in text above
                PES_Losses = PES_Losses - Num_PES_in_Packet_Before_Loss ;
            // change of timestamp in lost packet but PES was in packet before
        }
    }
};
```

If the RTP stream contains a bitstream with continuous timestamps, the RTP timestamp is not exploitable in this function. In this case, this process is replaced by an analysis of the video DTS (see below).

4.2.4.4.2.5. Loop over the TS Packets of one RTP-packet:

Input

StrippedUDPPackets, LostRTPPackets

Output

VideoFrames and/or AudioFrames

Process

After the RTP sequence number and the RTP timestamp have been examined, a loop over all TS packets contained in the current RTP packet (7 TS packets most of the time, but possible less) is performed, with the following steps.

- First of all the media stream of the current TS packet is detected by using its PID. The possible options for the media type of a TS packet are video, audio, or system.

- If the current TS packet contains a PES header, the index to the corresponding `VideoFrames` or `AudioFrames` array is incremented and it is denoted whether the end of the previous frame was lost or not: If there is a PES header and packets were lost and the `AssumedFrameEnd` variable was not set, a loss of the previous frame end (which may include complete losses of subsequent frames) can be assumed.

```
PrevEndLost = LostRTPPackets && (AssumedFrameEnd == 0) ;
PrevEndOK   = El_Stream->AssumedFrameEnd ;
```

- Depending on the detected media type of the TS packet under consideration different routines for video and audio are then executed as described below. System TS packets are ignored.

4.2.4.1-4.2.5.1. Estimate Number of lost TS packets

In principle, the continuity counter (CC), which is part of the TS header is used to estimate the number of lost TS packets, by calculating

```
LostTSPackets = (CC[n] - CC[n-1] - 1) & 0x0f // & 0x0f for the possible wrap around
```

Since the CC field of the MPEG-2 TS header consists of 4 bits, losses of more than 15 TS packets must be estimated. Since 16 TS packets need at least 3 RTP packets to be transmitted, an estimation is only necessary if more than 2 RTP packets in a row are lost. If this is the case, the number of "wraps around" has to be estimated ¹.

```
LostTSPackets += 16*NumWrapAround // NumWrapAround = 0,1,2 ...
```

To do so for video, the average number of TS packets in one RTP packet for the current media stream (audio or video) is calculated from statistical values of previous packets. It is then easy to calculate the average number of lost TS packets for the given number of currently lost RTP packets (`ExpectedAvLoss`), as shown below in the pseudocode. The multiples of 16 of this number imply a certain number of "wraps around" of the CC (`AvWrap`). If this average number of "wraps around" is assumed, the real number of TS packet losses can be calculated (`MeasuredAvLoss`). The same calculations are then done not with the average number of the lost TS packets, but with the maximum number (assuming one RTP-packet had contained 7 TS packets of one stream type). Finally, it is examined for which version the difference between measured and expected value is lower. The respective number of "wraps around" is then multiplied by 16 and added to `LostTSPackets`.

For audio, it is assumed that audio-packets are so sparsely placed in the RTP-packets that a "wrap-around" would not happen. A single wrap-around is only assumed if the `ExpectedAvLoss` is more than 8 packets larger than the measured value.

¹ Note: for the MPEG2-TS/UDP case the true number of lost TS packets had already been estimated in 4.2.3, to be able to estimate the number of lost UDP-packets. The resulting values could just be copied to this process. For the sake of simplicity, no distinction is made between MPEG2-TS/UDP and MPEG2-TS/RTP/UDP at this stage though.

```

if( LostRTPpackets > 2 )
{
TsPerRTP = Average_TSperRTP(); // calculate from statistics of previous Packets

    if( Media type of TS packet is video )
    {
TsPerRTP = Average_TSperPacket(VIDEO) ; // from statistics of previous Packets
MaxLoss = 7*LostRTPpackets;
MaxWraps = (LostRTPpackets<3)? 0 : (MaxLoss>>4)-(LostTSpackets > (MaxLoss & 0x0f)) ;
ExpectedAvLoss = int(TsPerRTP * LostRTPpackets + 0.5) ;
AvWraps = (ExpectedAvLoss >> 4) - (LostTSpackets > (ExpectedAvLoss & 0x0f)) ;
MeasuredAvLoss = LostTSpackets + (_AvWraps << 4) ;
ExpectedMaxLoss= 7*LostRTPpackets;
MaxWraps = (ExpectedMaxLoss>>4) - (LostTSpackets > (ExpectedMaxLoss & 0x0f)) ;
MeasuredMaxLoss= LostTSpackets + (MaxWraps << 4) ;

        OptWrap = ((abs(ExpectedAvLoss-MeasuredMaxLoss) <= abs(MaxLoss-MeasuredMaxLoss
                        || (AvWraps < (MaxWraps-1))) )? AvWraps : MaxWraps);
        LostTSpackets _= LostTSpackets + (OptWrap << 4) ;
    }
    else
    {
TsPerRTP = Average_TSperPacket(AUDIO) ; // from statistics of previous Packets
ExpectedAvLoss = int(TsPerRTP* LostRTPpackets + 0.5) ;
while( (ExpectedAvLoss-LostTSpackets) > 8 )
    LostTSpackets += 16 ;
};

    if( LostTSpackets > (int) LostRTPpackets*7 )
        LostTSpackets -= 16 ;
    }
}

```

4.2.4.2.4.2.5.2. Analyze video TS packet

Figure 3 illustrates the block diagram for the analysis of a video TS packet and the respective processes are further analysed below (c.f. module “Process Video TS Packet” in Figure 2).