ETSI TR 126 902 V11.0.0 (2012-10)

**Digital cellular telecommunications system (Phase 2+);
Universal Mobile Telecommunications System (UMTS);
LTE;
Video codec performance
(3GPP TR 26.902 version 11.0.0 Release 11)**

Reference
RTR/TSGS-0426902vb00

Keywords
GSM,LTE,UMTS

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Intellectual Property Rights

IPRs essential or potentially essential to the present document may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (http://ipr.etsi.org).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Foreword

This Technical Report (TR) has been produced by ETSI 3rd Generation Partnership Project (3GPP).

The present document may refer to technical specifications or reports using their 3GPP identities, UMTS identities or GSM identities. These should be interpreted as being references to the corresponding ETSI deliverables.

The cross reference between GSM, UMTS, 3GPP and ETSI identities can be found under http://webapp.etsi.org/key/queryform.asp.

Contents

Foreword

This Technical Report has been produced by the 3rd Generation Partnership Project (3GPP).

The contents of the present document are subject to continuing work within the TSG and may change following formal TSG approval. Should the TSG modify the contents of the present document, it will be re-released by the TSG with an identifying change of release date and an increase in version number as follows:

Version x.y.z

where:

- x the first digit:
  - 1 presented to TSG for information;
  - 2 presented to TSG for approval;
  - 3 or greater indicates TSG approved document under change control.
- y the second digit is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc.
- z the third digit is incremented when editorial only changes have been incorporated in the document.

1 Scope

The present document comprises a technical report on Video Codec Performance, for packet-switched video-capable multimedia services standardized by 3GPP.

2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication, edition number, version number, etc.) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies. In the case of a reference to a 3GPP document (including a GSM document), a non-specific reference implicitly refers to the latest version of that document *in the same Release as the present document*.

[1] IETF RFC 2429: "RTP Payload Format for the 1998 Version of ITU-T Rec. H.263 Video (H.263+)".

[2] IETF RFC 3550: "RTP: A Transport Protocol for Real-Time Applications", Schulzrinne H. et al, July 2003.

[3] ITU-T Recommendation H.263 (1998): "Video coding for low bit rate communication".

[4] 3GPP TS 26.110: "Codec for Circuit Switched Multimedia Telephony Service; General Description".

[5] 3GPP TS 26.111: "Codec for Circuit Switched Multimedia Telephony Service; Modifications to H.324".

[6] ITU-T Recommendation H.263 - Annex X (2004): "Annex X: Profiles and levels definition".

[7] ITU-T Recommendation H.264 (2003): "Advanced video coding for generic audiovisual services" | ISO/IEC 14496-10:2003: "Information technology - Coding of audio-visual objects - Part 10: Advanced Video Coding".

[8] ISO/IEC 14496-10/FDAM1: "AVC Fidelity Range Extensions".

[9] IETF RFC 3984: "RTP payload Format for H.264 Video".

[10] 3GPP TS 26.141: "IP Multimedia System (IMS) Messaging and Presence; Media formats and codecs".

[11] 3GPP TS 26.234: "Transparent end-to-end Packet-switched Streaming Service (PSS); Protocols and codecs".

[12] 3GPP TS 26.346: "Multimedia Broadcast/Multicast Service (MBMS); Protocols and codecs".

[13] 3GPP TS 26.235: "Packet switched conversational multimedia applications; Default codecs".

[14] 3GPP TS 26.236: "Packet switched conversational multimedia applications; Transport protocols".

[15] 3GPP TS 26.114: "IP Multimedia Subsystem (IMS); Multimedia telephony; Media handling and interaction".

[16] 3GPP TR 26.936: "Performance characterization of 3GPP audio codecs".

[17] 3GPP TR 25.101: "User Equipment (UE) radio transmission and reception (FDD)".

3 Abbreviations

For the purposes of the present document, the following abbreviations apply:

APSNR	Average PSNR
AVC	Advanced Video Codec
DCCH	Dedicated Control CHannel
DPDCH	Dedicated Physical Data CHannel
DTCH	Dedicated Traffic CHannel
HSPA	High-Speed Packet Access
IETF	Internet Engineering Task Force
IMS	Internet protocol Multimedia Subsystem
IP	Internet Protocol
MAC	Medium Access Control
MBMS	Multimedia Broadcast/Multicast Service
MSE	Mean Square Error
MTSI	Multimedia Telephony over IMS
NAL	Network Abstraction Layer
NSD	Normalized Square Difference
PANSD	PSNR of Average Normalized Square Difference
PDU	Protocol Data Unit
PDVD	Percentage of Degraded Video Duration
PSC	Packet-Switched Conversational
PSNR	Peak Signal-to-Noise Ratio
PSS	Packet-switched Streaming Service
RFC	IETF Request For Comments
RLC	Radio Link Control
RTCP	RTP Control Protocol
RTP	Real-time Transport Protocol
SDP	Session Description Protocol
TTI	Transmission Time Interval
UDP	User Datagram Protocol
UE	User Equipment
UTRAN	UMTS Terrestrial Radio Access Network

4 Document organization

The present document is organized as discussed below.

- Clause 5 introduces the service scenarios, including their relationship with 3GPP services. Furthermore, it discusses the performance measurement metrics used in the present document.
- Clause 6 (performance figures) defines representative test cases and contains a listing, in the form of tables, performance of video codecs for each of the test cases.
- Clause 7 (supplementary information on figure generation) contains pointers to accompanying files containing video sequences, anchor bit streams, and error prone test bit streams. It also describes the mechanisms used to generate the anchor compressed video data, compressed video data exposed to typical error masks, and descriptions on the creation of error masks.
- Annex A sketches one possible environment that could be used by interested parties as a starting point for defining a process to assess the performances of a particular video codec against the performance figures.
- Annex B introduces details on the H.263 encoder and decoder configurations.
- Annex C introduces details of the H.264 encoder and decoder configurations
- Annex D introduces details on the usage of 3G file format in the present document.
- Annex E introduces details on the usage of RTPdump format in the present document.

- Annex F introduces details on the simulator, bearers, and dump files.
- Annex G introduces the details on the Quality Metric Evaluation.
- Annex H introduces the details on the Video Test Sequences.
- Annex I provides information on verification of appropriate use of the tools provided in this document.

5 Service scenarios and metrics

Video transmission in a 3GPP packet-switched environment conceptually consists of an Encoder, one or more Channels, and a Decoder. The Encoder, as defined here, comprises the steps of the source coding and, when required by the service, the packetization into RTP packets, according to the relevant 3GPP Technical Specification for the service and media codec in question. The Channel, as defined here comprises all steps of conveying the information created by the Encoder to the Decoder. Note that the Channel, in some environments, may be prone to packet erasures, and in others it may be error free. In an erasure prone environment, it is not guarantied that all information created by the Encoder can be processed by the Decoder; implying that the Decoder needs to cope to some extent with compressed video data not compliant with the video codec standard. The Decoder, finally, de-packetizes and reconstructs the - potentially erasure prone and perhaps non-compliant - packet stream to a reconstructed video sequence. The only type of error considered at the depacketizer/decoder is RTP packet erasures.

5.1 Service scenarios

3GPP includes video in different services, e.g. PSS [11], MBMS [12], PSC [13], [14], and MTSI [15]. This report lists the performance figures only one service scenario focusing on an RTP-based conversational service such as PSC or MTSI.

- **Service scenario A (PSC/MTSI-like)** relates to conversational services involving compressed video data (an erasure prone transport, low latency requirements, application layer transport quality feedback, etc.). In this scenario, UE-based video encoding and decoding are assumed. The foremost examples for this service scenario are PSC or MTSI. Within the this service scenario, the performance of an encoder and a decoder is of importance for the service quality. Service scenario A refers to the performance of a decoder to consume a possibly non-compliant (due to transmission errors) compressed video data generated by an encoder that fulfils the provision of sufficient quality in this scenario.

5.2 Performance metrics

This clause defines performance metrics as used in section 6, to numerically and objectively express a Decoder's reaction to compressed video data that is possibly modified due to erasures. Only objective metrics are considered which can be computed from sequences being available in a 3G format as described in Annex D by using the method detailed in annex G.

The following section provides a general description of the quality metrics. For the exact computation with the availability of sequences in 3G format please refer to annex G.

The following acronyms are utilized throughout the remainder of this subclause:

- *OrigSeq*: The original video sequence that has been used as input for the video encoder.
- *ReconSeq*: The reconstructed video sequence, the output of a standard compliant decoder that operates on the output of the video encoder without channel simulation, i.e. without any errors. Timing alignment between the OrigSeq and ReconSeq are assumed.

- *ReceivedSeq*: The video sequence that has been reconstructed and error-concealed by an error-tolerant video decoder, after a) the video encoder operated on the OrigSeq and produced an error free packet stream file as output, b) the channel simulator used the error free packet stream file and applied errors and delays to it so to produce an error-prone packet file which is used as the input of the error-tolerant video decoder. For comparison purpose, a constant delay between OrigSeq and the ReceivedSeq is assumed, whereby this constant delay is removed before comparison.

Each of the following metrics generates a single value when run for a complete video sequence.

5.2.1 Average Peak Signal-to-Noise Ratio (APSNR)

The average Peak Signal-to-Noise Ratio (APSNR) calculated between all pictures of the *OrigSeq* and the *ReconSeq* or the *ReceivedSeq*, respectively. First, the Peak Signal-to-Noise Ratio (PSNR) of each picture is calculated with a precision sufficient to prevent rounding errors in the future steps. Thereafter, the PSNR values of all pictures are averaged. The result is reported with a precision of two digits.

NOTE: This is the traditional metric referred to as PSNR in the academic literature and in the context of video compression research.

Only the luminance component of the video signal is used.

In case that results from several *ReceivedSeq* are to be combined, the average of all PSNR values for all *ReceivedSeq* is computed as the final result.

5.2.2 PSNR of Average Normalized Square Difference (PANSD)

The PSNR of Average Normalized Square Difference (PANSD) is calculated between all pictures of the *OrigSeq* and the *ReceivedSeq*, respectively. First, the normalized square difference, also know as Mean Square Error (MSE) of each picture is calculated with a precision sufficient to prevent rounding errors in the future steps. Thereafter, the NSD values of all pictures are averaged. The result is reported with a precision of two digits. Then, a conversion of this value into a PSNR value is carried out.

Only the luminance component of the video signal is used.

In case that results from several *ReceivedSeq* are to be combined, the average of all NSD values for all *ReceivedSeq* is computed and the final result is the PSNR over this averaged NSD.

5.2.3 Percentage of Degraded Video Duration (PDVD)

The Percentage of Degraded Video Duration (PDVD) is defined as the percentage of time of the entire display time for which the PSNR of the erroneous video frames are more than x dB worse than PSNR of the reconstructed frames whereby x is set to 2 dB. This metric computation requires three sequences, the *OrigSeq*, the *ReconSeq*, and the *ReceivedSeq*.

Only the luminance component of the video signal is used.

In case that results from several *ReceivedSeq* are to be combined, the average of all PDVD values for all *ReceivedSeq* is computed as the final result.

6 Test case definition and performance figures

6.1 Preliminary remarks

The test cases defined in this clause represent typical or worst-case transmission scenarios for the evaluated 3GPP video services. In all cases the respective constraints of the service have been taken into account, especially in the selection of video encoding tools. Purposely, some well-known and recognized optimization features, for example for error resilience, have been excluded. It is obvious that these optimizations are possible and, at least to some extent, also expected in real-world implementations.

It is also *not* the intention of this document to provide a comparison of different video codecs.

6.2 Definitions

6.2.1 Video test sequences

The input to the encoding process is a source sequence in 3G file format. All applied test sequences are attached to the present document in zip-folder *TestSequences.zip*. Three different test sequences are provided.

Table 1: Video test sequences

Sequence	Name	File Name	Frame Rate
1	**Stunt**	**stunt_qcif.3gp**	**15 fps**
2	**Bar**	**bar-30s.3gp**	**12 fps**
3	**Party**	**lt-party-30s.3gp**	**12 fps**

Details on the test sequences can be found in annex H.

6.2.2 Encoding parameters

The video encoding process, the encoder is made aware of the service it is operating in and the expected bitrate. The only varying parameter in the considered scenario is the expected transmission bitrate in kbit/s. The encoder is not made aware of the expected error conditions.

The two considered bitrates are 64 kbit/s and 128 kbit/s. However, the encoder should take into account packetization overhead due to RTP/UDP/IP headers such that the actual encoding bitrate should be lower.

More detailed encoding parameters for each H.263 and H.264 are described in annexes B and C, respectively.

6.2.3 Transport parameters

The transport parameters define the bearer settings, the applied loss masks, and the number of statistical experiments. The definition of the bearer parameters is strongly related to the applied transport software.

The applied bearers and radio dump files are numbered from 1 to 8 (Bearer ID). In all cases, for statistical significance, 128 independent trials are applied.

For details of transport parameter definition we refer to clause 7.1.3 and annex F.

6.3 Service scenario A (PSC/MTSI-like service)

6.3.1 Encoding anchors

Encoding parameter combinations are defined by the following parameters:

- Anchor: Anchor number.
- Sequence: Sequence number and name.
- Bitrate: 64 kbit/s and 128 kbit/s.
- Codec: H.263 and H.264

The codec refers to the application of an encoder, which generates for:

- H.263 an RTP packet stream according to H.263 baseline (Profile 0) Level 45 [6] and RTP payload format for the ITU-T Recommendation H.263 video codec [3] in RFC 2429 [1].

- H.264 (AVC) Baseline Profile at Level 1b [8] and RTP payload format for the ITU-T Recommendation H.264 (AVC) video codec [7] is specified in IETF RFC3984 [9], where only the single NAL unit packetization mode is used.

Table 2 contains the encoding anchors for service scenario A.

Table 2: Encoding anchors for service scenario A

Encoding Anchor	Codec	Sequence	Bitrate
AA1-263	**H.263**	**1 Stunt**	**64 kbit/s**
AA2-263	**H.263**	**1 Stunt**	**128 kbit/s**
AA3-263	**H.263**	**2 Bar**	**64 kbit/s**
AA4-263	**H.263**	**2 Bar**	**128 kbit/s**
AA5-263	**H.263**	**3 Party**	**64 kbit/s**
AA6-263	**H.263**	**3 Party**	**128 kbit/s**
AA1-264	**H.264**	**1 Stunt**	**64 kbit/s**
AA2-264	**H.264**	**1 Stunt**	**128 kbit/s**
AA3-264	**H.264**	**2 Bar**	**64 kbit/s**
AA4-264	**H.264**	**2 Bar**	**128 kbit/s**
AA5-264	**H.264**	**3 Party**	**64 kbit/s**
AA6-264	**H.264**	**3 Party**	**128 kbit/s**

6.3.2 Test cases

The definition of each test case consists of:

- Test: Test case number.
- Encoding Anchor according to table 2.
- Transport Bearer: Bearer ID, as explained in section 6.2.3.

The test case definitions for service scenario A are collected in table 3.

Table 3: Test case definitions for service scenario A

Test Case	Encoding Anchor	Transport Bearer
TA01-263	**AA1-263**	**1**
TA02-263	**AA1-263**	**2**
TA03-263	**AA1-263**	**3**
TA04-263	**AA1-263**	**4**
TA05-263	**AA2-263**	**5**
TA06-263	**AA2-263**	**6**
TA07-263	**AA2-263**	**7**
TA08-263	**AA2-263**	**8**
TA09-263	**AA3-263**	**1**
TA10-263	**AA3-263**	**2**
TA11-263	**AA3-263**	**3**
TA12-263	**AA3-263**	**4**
TA13-263	**AA4-263**	**5**
TA14-263	**AA4-263**	**6**
TA15-263	**AA4-263**	**7**
TA16-263	**AA4-263**	**8**
TA17-263	**AA5-263**	**1**
TA18-263	**AA5-263**	**2**
TA19-263	**AA5-263**	**3**
TA20-263	**AA5-263**	**4**
TA21-263	**AA6-263**	**5**
TA22-263	**AA6-263**	**6**
TA23-263	**AA6-263**	**7**
TA24-263	**AA6-263**	**8**
TA01-264	**AA1-264**	**1**
TA02-264	**AA1-264**	**2**
TA03-264	**AA1-264**	**3**
TA04-264	**AA1-264**	**4**
TA05-264	**AA2-264**	**5**
TA06-264	**AA2-264**	**6**
TA07-264	**AA2-264**	**7**
TA08-264	**AA2-264**	**8**
TA09-264	**AA3-264**	**1**
TA10-264	**AA3-264**	**2**
TA11-264	**AA3-264**	**3**
TA12-264	**AA3-264**	**4**
TA13-264	**AA4-264**	**5**
TA14-264	**AA4-264**	**6**
TA15-264	**AA4-264**	**7**
TA16-264	**AA4-264**	**8**
TA17-264	**AA5-264**	**1**
TA18-264	**AA5-264**	**2**
TA19-264	**AA5-264**	**3**
TA20-264	**AA5-264**	**4**
TA21-264	**AA6-264**	**5**
TA22-264	**AA6-264**	**6**
TA23-264	**AA6-264**	**7**
TA24-264	**AA6-264**	**8**

6.3.4 Performance figures

Performance is reported as the response of a performance metric to a sequence of pictures reconstructed by a decoder. For each test case, results generated for the metrics are reported.

The video codec performance figures for service scenario A are collected in table 4.

Table 4: Video codec performance figures for service scenario A

Test Case	APSNR	PANSD	PDVD
TA01-263	**27.34 dB**	**27.00 dB**	**0**
TA02-263	**25.34 dB**	**23.29 dB**	**26.17**
TA03-263	**23.80 dB**	**21.51 dB**	**41.75**
TA04-263	**22.53 dB**	**20.38 dB**	**54.72**
TA05-263	**30.46 dB**	**29.90 dB**	**0**
TA06-263	**27.97 dB**	**24.84 dB**	**24.46**
TA07-263	**25.75 dB**	**22.40 dB**	**42.37**
TA08-263	**23.89 dB**	**21.00 dB**	**54.95**
TA09-263	**30.12 dB**	**30.02 dB**	**0**
TA10-263	**28.79 dB**	**28.21 dB**	**23.83**
TA11-263	**27.69 dB**	**26.93 dB**	**45.46**
TA12-263	**26.71 dB**	**25.95 dB**	**62.90**
TA13-263	**33.13 dB**	**33.03 dB**	**0**
TA14-263	**31.19 dB**	**30.02 dB**	**31.10**
TA15-263	**29.41 dB**	**28.10 dB**	**57.71**
TA16-263	**27.99 dB**	**26.79 dB**	**72.79**
TA17-263	**27.73 dB**	**27.37 dB**	**0**
TA18-263	**25.71 dB**	**24.02 dB**	**30.71**
TA19-263	**24.01 dB**	**22.17 dB**	**50.61**
TA20-263	**22.73 dB**	**20.98 dB**	**57.50**
TA21-263	**30.68 dB**	**30.34 dB**	**0**
TA22-263	**27.96 dB**	**25.20 dB**	**31.99**
TA23-263	**25.63 dB**	**22.90 dB**	**49.23**
TA24-263	**23.66 dB**	**21.29 dB**	**53.63**
TA01-264	**28.38 dB**	**27.76 dB**	**0**
TA02-264	**26.06 dB**	**23.64 dB**	**26.88**
TA03-264	**24.35 dB**	**21.86 dB**	**42.58**
TA04-264	**22.98 dB**	**20.66 dB**	**53.87**
TA05-264	**31.96 dB**	**31.43 dB**	**0**
TA06-264	**28.98 dB**	**25.01 dB**	**25.06**
TA07-264	**26.59 dB**	**22.59 dB**	**44.28**
TA08-264	**24.43 dB**	**21.03 dB**	**54.34**
TA09-264	**30.79 dB**	**30.71 dB**	**0**
TA10-264	**29.54 dB**	**28.77 dB**	**22.89**
TA11-264	**28.43 dB**	**27.41 dB**	**42.51**
TA12-264	**27.49 dB**	**26.45 dB**	**57.92**
TA13-264	**34.34 dB**	**34.26 dB**	**0**
TA14-264	**32.48 dB**	**30.84 dB**	**26.97**
TA15-264	**30.73 dB**	**28.80 dB**	**50.23**
TA16-264	**29.23 dB**	**27.41 dB**	**67.85**
TA17-264	**28.16 dB**	**27.83 dB**	**0**
TA18-264	**26.12 dB**	**24.16 dB**	**31.33**
TA19-264	**24.41 dB**	**22.21 dB**	**54.64**
TA20-264	**23.18 dB**	**21.07 dB**	**69.25**
TA21-264	**31.65 dB**	**31.33 dB**	**0**
TA22-264	**28.86 dB**	**25.38 dB**	**33.39**
TA23-264	**26.50 dB**	**23.01 dB**	**58.32**
TA24-264	**24.46 dB**	**21.37 dB**	**75.46**

7 Performance figure generation

The performance figures in the present document have been generated according to the rationale discussed in this clause. A file-based approach has been employed. In the following drawings, files are indicated by using cylinder shapes, and functional modules operating on files (encoder, decoder, transport simulator, quality assessment) are depicted using rounded rectangles. Also, the interfaces in 3G file format and RTP dump are marked explicitly in the diagrams.

7.1 Performance figure generation for service scenario A

7.1.1 Overview

The service scenario A relates to the performance of a video codec in a conversational-like environment, and especially the performance of an error-tolerant decoder can be assessed, when processing video data that has been exposed to erasures stemming from the use of the 3GPP Transport Simulator.

For the generation of the performance figures, the process as depicted in figure 1 is applied. The process assumes the availability of a Compressed Video in RTP format representing the encoding anchor. The generation of the encoding anchors is discussed in subclause 7.1.2.

The encoding anchor for test case according to table 3 is exposed to RTP packet erasures and delays by the 3GPP Transport Simulator. The latter is being controlled by command line configuration information, configuration files, and error pattern files, all summarized as Bearer Number in subclause 7.1.3. The Transport Bearer number assigned in table 3 assigned to respected test case has been applied.

The result of this process is Erasure-prone compressed video in RTP format referred to as *Decoding Anchor*. This file is being reconstructed by an error-tolerant video decoder to generate a possibly error-prone video *ReceivedSeq* in the 3GP file format. Note that the video decoder use the RTP timestamps to reconstruct the presentation times. The presentation timestamps are included in the erroneous video sequence *ReceivedSeq*.

The quality assessment uses the original video source *OrigSeq* (in .3GP format), the reconstructed video file before transmission *ReconSeq* (in .3GP format) as well as the reconstructed error-prone video *ReceivedSeq* (in .3GP format). The *ReconSeq* is necessary such that the quality evaluation tool understands if video frames have been dropped intentionally by the decoder of it has been lost due to transmission errors. It is also necessary to compute the PDVD.

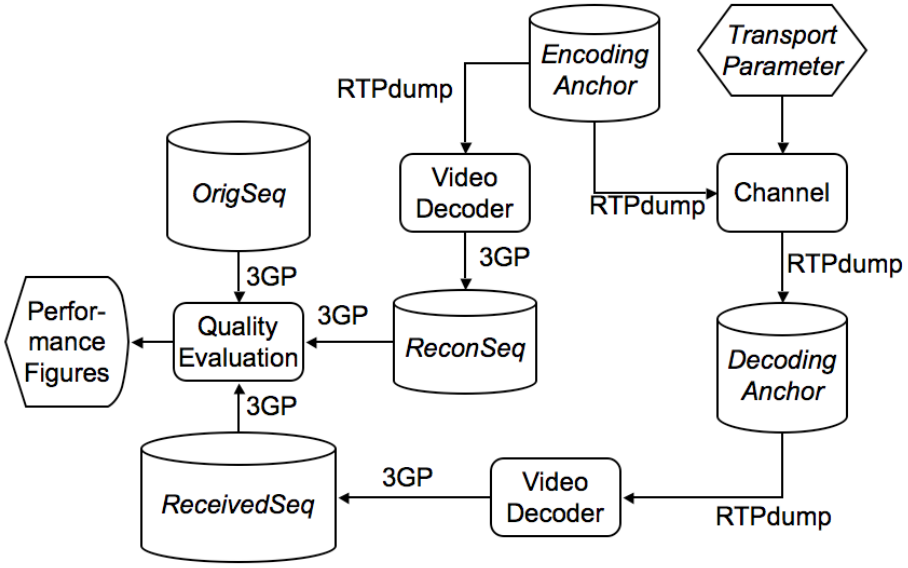

Figure 1: Environment for the generation of video performance figures for service scenario A

7.1.2 Encoding anchors

The encoding anchors are available in the attached file *EncodingAnchors_711.zip*.

The generation of the encoding anchor is shown is sketched in figure 2. The original sequence *OrigSeq* in .3GP file format is processed by a video Encoder (H.263 or H.264), controlled by the Encoder Parameters which are the service type and the bitrate. The result of this process is a Compressed Video packet stream file, conforming to the respective video compression standard, H.263 baseline or H.264 constrained baseline, the relevant RTP packetization, and the RTPdump file format as discussed in annex E. The bitstreams are encoded such that they match to the service constraints, namely low-complexity encoding as expected to run on a UE, low-delay rate control, and basic error resilience to support some extent of packet loss. More details on encoder settings for H.263 and H.264 are provided in annexes B and C, respectively.

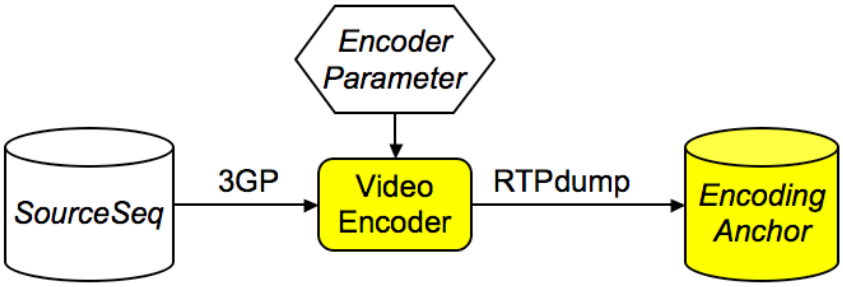

Figure 2: Generation of encoding anchor

The encoding anchors for scenario A are included in the archive *EncodingAnchors_711.zip* which is a attached to the present document.

The anchors are in RTPdump format, for details on the RTPdump format see annex E.

The naming conventions are as follows: <encoding_anchor>.rtp

encoding_anchor: according to the definition in table 2 as AA1-263, …, AA6-264.

7.1.3 3GPP transport simulator

The 3GPP Transport Simulator is attached to this document in archive *SA4Simulator_711.zip*. Error traces from UTRAN-like packet-switched dedicated bearers have been used for generating the video performance figures, shared channels such as HSPA have not been considered.

The command line parameters to the 3GPP Transport Simulator are, the filenames of the Compressed Video in RTP format (for the performance generation this corresponds to the respective encoding anchor) and the Erasure Prone Video in RTP format (corresponds to the decoding anchor), and the Bearer ID (an integer), and the initialization of the Random Number Generator (Random Seed).

The Bearer configuration could be found in file bearers.txt and has been set as follows:

```
# This file contains some bearer configuration. The bearers can be indexed by the number.
# The specific columns are explained in the following
# Number:    Number of the bearer used as index (integer)
# File:    File name of the error masks, can be bit errors or packet errors
# Format:    Gives the format of the file (binary for bit errors, ascii for packet errors)
# TTI:     Transmission Time Interval in ms
# RFS:     Radio Frame Size in bytes describes the RLC-PDU size
# note that 8*RFS/TTI results in the bit rate in kbit/s
# Mode:    Transmission Mode: UACK is unacknowledged bearer, ACKP is acknowledged bearer with
persistent mode, ACKN non-persistent, FECK is unacknowledged and FEC
# System:    CDMA2000, UMTS, GPRS, EGPRS, main difference is in sizes of fields added for headers
# CRUIH:     Compressed RTP/IP/UDP header size assuming header compression
# RDel:    (only for ACK mode) The retransmission delay before it is available at the encoder in
multiples of the TTI
# NoRet:     (only non-persistent ACK mode ACKN) Number of Retransmission for ACK mode
#
# The following bearers are defined
# Number    File                                Format  TTI     RFS     Mode System CRUIH
```

```
# PSC Bearers
# 64 kbit/s
1       0                                       iid     20      160     UACK    UMTS    5
2       PSC__64kbps_20ms_BLER_0_5.txt           ascii   20      160     UACK    UMTS    5
3       PSC__64kbps_20ms_BLER_1_0.txt           ascii   20      160     UACK    UMTS    5
4       PSC__64kbps_20ms_BLER_1_5.txt           ascii   20      160     UACK    UMTS    5
# 128 kbit/s
5       0                                       iid     20      160     UACK    UMTS    5
6       PSC__128kbps_20ms_BLER_0_5.txt          ascii   20      320     UACK    UMTS    5
7       PSC__128kbps_20ms_BLER_1_0.txt          ascii   20      320     UACK    UMTS    5
8       PSC__128kbps_20ms_BLER_1_5.txt          ascii   20      320     UACK    UMTS    5
```

NOTE: Software usage requires the above bearer settings to be put in ./Config/bearers.txt relative to the executable.

The following configuration for the channel simulator has been used:

```
RTPinfile       = filename of input file
RTPoutfile      = filename of output file
LogFile         = filename of logfile
StatFile        = filename of statistics file
Bearer          = see below
RandomSeed      = 1-128
ErrorFreeRTP    = 4
TSModeSender    = 0   # 0 use TS
MaxSendingDelay = 0   # 0 ignore TS
MaxE2EDelay     = 500 # 0 ignore TS, > 0 drop packet at receiver if delayed
```

NOTE: The configuration is put in a text file and indicated to the software by `-f` argument (for example below in "Usage of simulator"). Software usage requires at least one space or tab character before any line ending in this file.

With this configuration all packets arriving later than 500 ms compared to the time they were generated, are being removed from the Erasure Prone Video RTP file. Similarly, all packets containing which would have been mapped on a erroneous RLC/MAC-frame are being removed. The first four RTP packets are excluded from the error simulation, so to ensure that the parameter sets and the first intra slice are available for processing at the H.263 and H.264 decoder.

NOTE: This optimization is justified as H.264 parameter sets, in a real-world environment, are made available to the decoder through the session negotiation process and the delay of the initial I-frame should not harm the decoding.

Usage of simulator:

```
sa4sim -f psc.cfg -p RTPinfile=<user defined> -p Bearer=<1-8> -p RandomSeed=<1-128>
```

7.1.4 Decoding anchors

The decoding anchors are not included in the present document. However, the anchors can be easily generated by applying the encoding anchors to the simulator.

EXAMPLE: To generate the decoding anchor for test case TA01-263 with random seed 1, run the simulator as follows.

```
sa4sim -f psc.cfg -p RTPinfile="AA1-263.rtp" RTPoutfile="TA01-263_1.rtp" -p Bearer=1 -p RandomSeed=1
```

The proposed naming conventions are as follows: <testcase>_< seed>.rtp

testcase: according to the definition in table 3 as TA01-263, …, TA24-264.

seed: Random seed of the simulator from 1, …, 128.

7.1.5 Error-tolerant video decoder

The error tolerant Video Decoders are compliant with the respective video coding standards (H.263 or H.264) when receiving and handling compliant input streams. However, when non-compliant input data is received, they implement additional mechanisms for error tolerance and error concealment. In summary, the error-tolerant video decoders utilized to generate the video codec performance figures:

- Do not crash when receiving non-compliant streams.

- Employ the so-called "last picture copy" error concealment technique, when the loss of pictures or picture parts is detected. That is, missing macroblocks are copied from the spatially co-located macroblocks in the previous reconstructed picture.

- Are capable of gracefully handling lost pictures in that they generate timing information in the Reconstructed Error Prone video file that indicates the lost picture. The applied decoders write the presentation time of all fully or partly received pictures in the 3G output files. For all completely lost pictures, the 3G output file does not necessarily contain any picture as the detection of lost pictures is non-trivial. However, the timing information of received pictures along with the usage of the *ReconSeq* in the quality evaluation allows the computation of all quality metrics.

More details on decoders can be found in annex B (for the H.263 decoder) and annex C (for the H.264 decoder) respectively.

Annex A:
Performance assessment of a codec implementation

This annex gives some guidelines how the content of the present document can be used to assess the performance of a codec implementation.

A.1 Decoder performance assessment

The performance of an implemented video decoder with respect to the provided performance figures can be assessed.

In this case a similar setup as shown in figure 1 is suggested to be used. However, instead of using the reference video decoder, the decoder under assessment is used when applied to the streams being exposed to errors. The modification is shown in figure A.1.

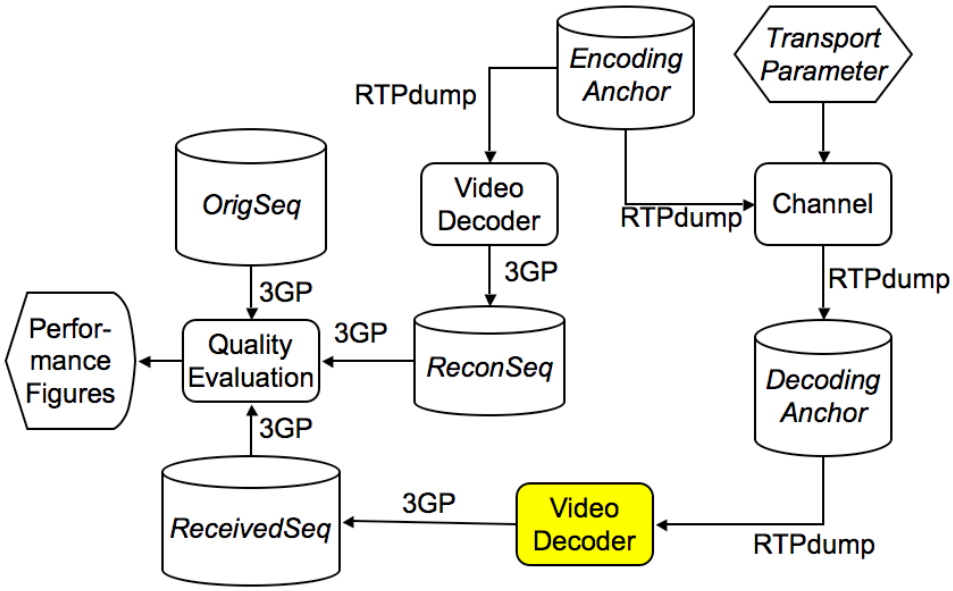

Figure A.1: Setup for video decoder assessment

A.2 Encoder performance assessment

This test intends checking the performance of an implemented video encoder against the provided performance figures. A reference decoder reconstructs the output of implemented encoder. The following procedure may be followed.

Figure A.2 shows the use of setup for video encoder assessment for scenario A. To assess the video encoder in service scenario A, an *Encoding Stream* is generated by applying the setup in figure 1, but instead of the reference encoder, the encoder under assessment is applied. The encoder should encode to match the service bitrate.

The generated encoded stream is highlighted in figure A.2. This stream is exposed to the channel by applying the transport parameters, but also the 3G file is reconstructed without application to the channel, but back-to-back usage of the decoder. In both cases a reference decoder is applied to generate a reconstructed 3GP video sequence and an error prone 3GP video sequence. Along with the original sequence, these two sequences are assessed applying the quality evaluation tool.

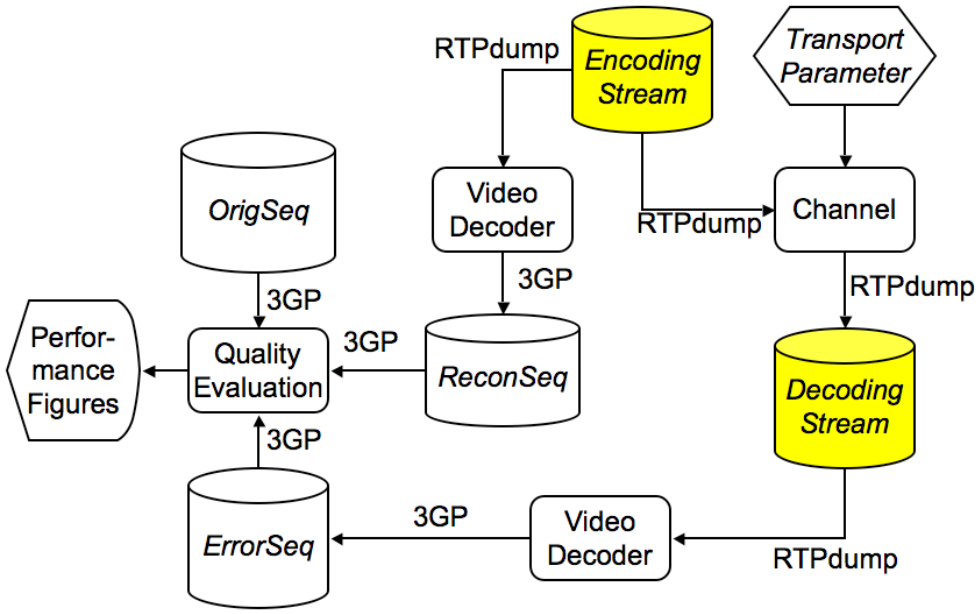

Figure A.2: Setup for video encoder assessment for scenario A

Annex B:
H.263 Codec description

B.1 Decoding process

The H.263 decoder used for the performance figure generation:

- Allowed processing RTPdump files.
- Employed "last picture copy" error concealment technique,i.e. when the loss of pictures or picture parts is detected. That is, missing macroblocks are copied from the spatially co-located macroblocks in the previous reconstructed picture.

B.2 Encoding process

The high-level H.263 encoder settings for the generation of the anchors were as follows:

- H.263 "baseline" mode of operation is employed, without the use of any optional enhancements.
- Constant bitrate encoding, frame skips not allowed.
- Pseudorandom intra macroblock refresh rate of 5%.

Annex C:
H.264 Codec description

C.1 Decoding process

The H.264 decoder used for the performance figure generation:

- Allowed processing RTPdump files.
- Employed "last picture copy" error concealment technique,i.e. when the loss of pictures or picture parts is detected. That is, missing macroblocks are copied from the spatially co-located macroblocks in the previous reconstructed picture.

C.2 Encoding process

The high-level H.264 encoder settings for the generation of the anchors were as follows:

- Single, (temporally) most recent reference frame for motion compensation.
- Only 16x16, 16x8, 8x16 and 8x8 inter-prediction modes allowed.
- Constant bitrate encoding, frame skips not allowed.
- Pseudorandom intra macroblock refresh rate of 5 %.
- In-loop deblocking filter enabled.

Annex D:
3GPP file format extension for raw video

The raw video sequences are stored in a format such that display timing is maintained. The 3GP file format was viewed as most appropriate. Attached to this document is a package *ISOFileFormatConverter_711.zip* which allows conversion of yuv video files to and from 3GPP file format. This package contains the relevant library as well as sample applications.

An alternate software is the reference code for the 3GPP file format (indeed, all ISO file formats), which can be obtained on request from the -MP4' registration authority at http://www.mp4ra.org. Some explanation on how to use the reference software for the same purpose (conversion of yuv video files and 3GPP file format) is provided in the following.

Raw video is stored as samples in a video track in ISO Base Media Format family files (such as MP4, 3GP and so on). That is, it uses the same video handler, video media header, etc., as a video track containing, for example, H.263. Raw video can take various formats - based on choice of color model, sub-sampling, and so on. As is usual in ISO files, the format of the video (the -decoder' needed) is declared by the sample entry 4-character-code. For YUV 4:2:0 video the 4-character code is -j420' (jay four two zero) (this happens to match QuickTime's current 4CC for 4:2:0).

Many video formats add boxes to the sample entry to parameterize the video (e.g. an AVC configuration box for AVC video); this format adds no boxes.

Note that the sample size table can use the compact form (constant sample size), storing only one value. Similarly the time-to-sample table can run-length compress to a single value for the input files (which have a constant frame rate).

For the use in 3GPP testing, the timescale of the media should match the movie timescale. A value of 600 is used. Likewise, the sample entry dimensions must record the image size (dimensions of the Y array), and these dimensions should be reflected in the track and movie dimensions.

Reference code for the 3GPP file format (indeed, all ISO file formats) can be obtained on request from the -MP4' registration authority at www.mp4ra.org. The source code below shows a sample program which converts a YUV file at a constant frame rate into a 3GPP file. It can also make a QuickTime movie file; in this case, QuickTime can be used to play the file, which may be advantageous. (QuickTime does not currently support opening 3GP files containing raw video.)

The example code from the registration authority contains a sample program for reading files, also.

A complete specification for the structure of 3GPP files may be obtained as a freely available standard from ISO (www.iso.ch); the standard is ISO/IEC 14496-12:2005.

```
/*

    sample program to build raw YUV 4:2:0 QuickTime (or 3GP) files using the file
    format  reference software

    Dave Singer, May 2006

*/
#include "ISOMovies.h"

#define infile "foreman_QCIF.yuv"
#define outfile "foreman_QCIF.mov"
#define x_width 176
#define y_height 144
#define bytes_per_frame ((x_width * y_height * 3)/2)
#define timescale 600
#define frameduration 20


MP4Err createMyMovie( char *filename );
MP4Err addMySamples( MP4Track theTrack, MP4Media theMedia, MP4Movie moov, char* fromfile );

int main( int argc, char **argv )
{
    MP4Err err;
```

```
    err = createMyMovie( outfile );
    fprintf( stderr, "createMyMovie returns %d\n", err );
    return err;
}

MP4Err createMyMovie( char *filename )
{
    MP4Err err;
    MP4Movie moov;
    MP4Track trak;
    MP4Media media;
    u64 mediaDuration;

    err = MP4NoErr;
//  err = New3GPPMovie( &moov, 6 ); if (err) goto bail;
    err = QTNewMovie( &moov ); if (err) goto bail;

    err = MP4NewMovieTrack( moov, MP4NewTrackIsVisual, &trak ); if (err) goto bail;
    err = MJ2SetTrackDimensions(    trak, x_width<<16, y_height<<16 );
    err = MP4NewTrackMedia( trak, &media, MP4VisualHandlerType, timescale, NULL );
        if (err) goto bail;
    err = MP4BeginMediaEdits( media ); if (err) goto bail;
    err = addMySamples( trak, media, moov, infile ); if (err) goto bail;

    err = MP4EndMediaEdits( media ); if (err) goto bail;
    err = MP4GetMediaDuration( media, &mediaDuration ); if (err) goto bail;
    err = MP4InsertMediaIntoTrack( trak, 0, 0, mediaDuration, 1 ); if (err) goto bail;

    err = ISOWriteMovieToFile( moov, filename ); if (err) goto bail;
    err = MP4DisposeMovie( moov ); if (err) goto bail;
bail:
    return err;
}


MP4Err addMySamples( MP4Track trak, MP4Media media, MP4Movie moov, char* the_file )
{
    MP4Err err;
    MP4Handle sampleEntryH;
    MP4Handle sampleDataH;
    MP4Handle sampleDurationH;
    MP4Handle sampleSizeH;
    u32 first_sample;

    FILE* fd;

    fd = fopen( the_file, "r" );

    err = MP4NoErr;
    err = MP4SetMediaLanguage( media, "und" ); if (err) goto bail;

    err = MP4NewHandle( 0, &sampleEntryH ); if (err) goto bail;
    err = MP4NewSampleDescription( trak, sampleEntryH,
        1,
        0,
        0,
        0,
        0,
        0,
        NULL ); if (err) goto bail;
    err = ISOSetSampleDescriptionDimensions( sampleEntryH, x_width, y_height );
        if (err) goto bail;
    err = ISOSetSampleDescriptionType( sampleEntryH,
        MP4_FOUR_CHAR_CODE( 'j', '4', '2', '0' ) ); if (err) goto bail;

    err = MP4NewHandle( sizeof(u32), &sampleDurationH ); if (err) goto bail;
    *((u32*) *sampleDurationH) = frameduration;

    err = MP4NewHandle( bytes_per_frame, &sampleDataH ); if (err) goto bail;
    err = MP4NewHandle( sizeof(u32), &sampleSizeH ); if (err) goto bail;
    * ((u32 *) (*sampleSizeH)) = bytes_per_frame;

    first_sample = 1;

    for ( ;; )
    {
        int read_count;
```

```
        read_count = fread( *sampleDataH, 1, bytes_per_frame, fd );
        if (read_count < bytes_per_frame) break;

        err = MP4AddMediaSamples( media, sampleDataH, 1,
            sampleDurationH, sampleSizeH,
            ( first_sample ? sampleEntryH : NULL), NULL, NULL );
        if (err) goto bail;
        first_sample = 0;
    }

    if ( sampleEntryH )
    {
        err = MP4DisposeHandle( sampleEntryH ); if (err) goto bail;
        sampleEntryH = NULL;
    }

    fclose(fd);

bail:
    return err;
}
```

Annex E:
RTPDUMP file format

It was agreed that it is essential to maintain timing information with media packets. The rtpdump file format is used as it fulfils all the requirements.

The rtpdump file format has been originally proposed by Henning Schulzrinne, see http://www.cs.columbia.edu/IRT/software/rtptools/. Within the scope of this report, only the binary version of the file format is of relevance. The file is constructed as follows:

The file starts with one line of ASCII coded text, indicating:

```
#!rtpplay1.0 address/port\n
```

wherein "address" stands for an IP address (e.g. 192.168.1.2) and port stands for a port number, e.g. 1234. Neither value is used by the toolchain employed in this report. "\n" stands for carriage return/linefeed.

The ASCII header is followed by one binary header (RD_hdr_t) and one RD_packet_t structure for each received packet. All fields are in network byte order. The RTP and RTCP packets are recorded as-is.

```
typedef struct {
  struct timeval start;  /* start of recording (GMT) */
  u_int32 source;        /* network source (multicast address) */
  u_int16 port;          /* UDP port */
} RD_hdr_t;

typedef struct {
  u_int16 length;    /* length of packet, including this header (may
                        be smaller than plen if not whole packet recorded) */
  u_int16 plen;      /* actual header+payload length for RTP, 0 for RTCP */
  u_int32 offset;    /* milliseconds since the start of recording */
} RD_packet_t;
```

Annex F:
Simulator and bearer details

F.1 Simulator package

The simulator and the corresponding error masks are attached to this document in archive *SA4Simulator_711.zip*. The simulation tool has been used for different purposes in 3GPP SA4, e.g. in 3GPP TR 26.936 [16] for the performance verifications in error conditions, but some modifications and updates have been done to fulfil the requirements of this work.

F.2 Simulator description

The processing chain which was used for these video performance assessment aimed at resembling a real-world transmission scenario. This includes source encoding and RTP packetization for the part of the media. The simulator performance mapping of RTP/IP packets to RLC-PDUs, de-packetization and decoding. Error insertion was performed on the RLC-PDU level using a file-based approach and simulator software for UTRAN which is briefly described in the following. In principle, all RTP packets which were affected at least partly by a RLC-PDU error were discarded. The simulation also modeled the application of Header Compression by reducing the RTP/UDP/IP packet headers to some constant number of bytes. All packet loss rates in the present document mentioned refer to RLC-PDU packet loss rates, not to RTP packet loss rates.

The transport simulation tool was used to map RLC-PDU error masks and bearer parameters (bitrates, RLC-PDU sizes, transmission time intervals) to RTP packet losses and. A block diagram as well as the corresponding interfaces of this software are shown in figure C.1.

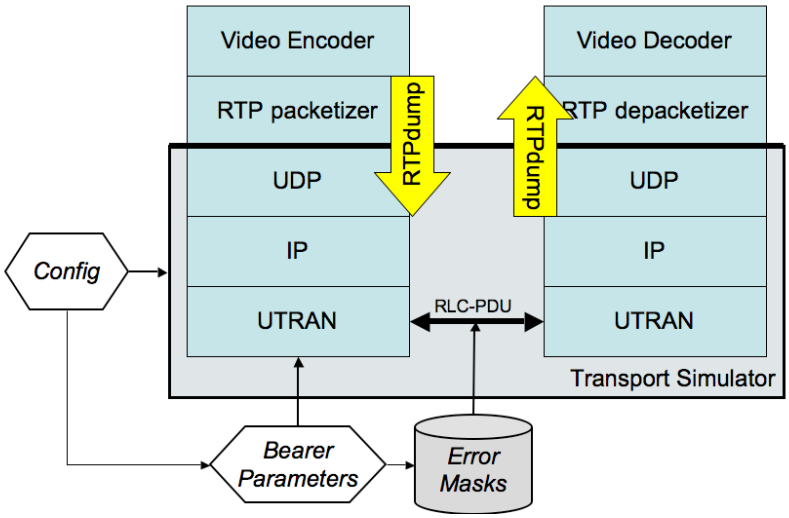

Figure C.1: Transport simulator: block diagram and interfaces

The video codecs produce RTP packets which are encapsulated into RTPdump format. The resulting packet stream is stored in RTPdump and can be processed by the simulator. The RTPdump header includes the size of the RTP packet including payload and header in bytes and a time stamp. This time stamp is used for different purposes: It basically indicates when the included RTP packet is virtually available for the next processing unit. This allows the video encoder to signal to the simulation software when the packet is available and also the simulation software when the packet is available to the decoder.

In addition to the modification of the time stamp, the software tool maps the RTP packets to the error masks on RLC-PDU layer. One PDU error results in that all RTP packets which are partially or completely mapped to the lost PDU, to be discarded. Discarded packets are not further signaled to the decoder, but are just not written to the output file. All non-discarded packets are error-free. It is expected that the video decoder including the de-packetizer is able to detect packet losses by only employing the information being included in correctly received RTP packets.

The software tool can be configured by different parameter settings. An exemplary configuration file `psc.cfg` is provided. In the following we will briefly define the input parameters:

```
RTPinfile       = filename of input file
RTPoutfile      = filename of output file
LogFile         = filename of logfile
StatFile        = filename of statistics file
Bearer          = see below
RandomSeed      = 1-128
ErrorFreeRTP    = 4
TSModeSender    = 0   # 0 use TS
MaxSendingDelay = 0   # 0 ignore TS
MaxE2EDelay     = 500 # 0 ignore TS, > 0 drop packet at receiver if delayed
```

The configuration parameters are explained in more details in the following.

RTPinfile:

File name for the input file with RTPdump format. The timing information indicates the time instant when the packet is released by the encoder/RTP packetizer. The simulator can use this information to maintain a certain sending timeline and to send dummy data or data from other applications in case no data is available from the considered video stream or to drop video stream packets in case of buffer overflow at the transmitter. This allows for example the simulation of live encoding. The use of this information is optional and is indicated by TSSenderMode (see below).

RTPoutfile:

File name for output file also in RTPdump format. It is identical to the input file except that

- Entire RTP packets including length information, timing information, and payload might be missing in case that the RTP packet has at least partly been mapped to a lost RLC-PDU.
- The timing is altered such that the time instant is provided when the RTP packet is released by the receiver.

Bearer:

Logs events in the simulator.

Bearer:

A specific bearer can be selected using a number which addresses a bearer. The bearer is further specified in a file named <bearers.txt>. In this file each non-commented line (comment is #) represents a bearer. Additional bearers can be added. The bearer file is shown in subclause 7.1.3.

RandomSeed:

Integer value to modify the starting position in the error pattern. For longer simulations it is proposed to start with 0 and increment the value by 1 for each run. Furthermore, this value is used as the initial seed of the random generator.

ErrorfreeRTP:

Specifies a certain number of RTP Packets at the beginning of the file which can be forwarded directly to the receiver without being lost. This is especially important if for example the first packet contains setup information, and was used as specified in subclause 7.1.3.

TSModeSender:

- 0, i.e. the program evaluates the timestamp and does not send the packet until the internal clock is as least as high as the time stamp of the packet. Therefore, one can simulate live encoding.

MaxSendingDelay

Set to 0, the transmitter buffer is assumed to have infinite length.

Max E2Edelay

The maximum delay of a packet in ms between the time it was generated and it is released by the simulator. If this time is exceeded, the packet is dropped, i.e. consider as a late losses. If set to 0, no late losses are introduced.

Stat File:

Provides some information for each run. In detail, the following information is provided:

- Date and time.
- Bearer number.
- Starting position in error file determined by RandomSeed.
- Bit error rate for this transmission.
- Native RLC-PDU loss rate.
- Effective bit rate counting only correctly received RLC blocks.
- Total Number of RLC frames.
- Total Number of retransmitted RLC frames.
- Total Number of dummy RLC frames.
- Total Number of RTP packets excluding the first error free RTP packets.
- Total RTP packet loss rate.
- The bit rate for the video in kbit/s.
- The total amount of time to transmit this file in ms.

F.3 Error masks

RLC-PDU error masks for UTRAN were generated using the following parameters

- RLC-PDU loss rates: 0.5 %, 1, and 1.5 %.
- Bearer bitrates: 64 kbps and 128 kbps.
- Geometry: -3dB.
- Channel model: Vehicular-A, 3 km/hr.
- TTI: 20 ms.

It was assumed that the PDU sizes were 160 bytes and 320 bytes for 64 kbps and 128 kbps bearers, respectively. The channel model used in the simulation was case 2 channel from 3GPP TR 25.101 [17]. The channel profile is given in table F.1. The physical channel used was DPDCH. Both inner loop and outer loop power control were enabled. DTCH and DCCH were physically transmitted on DPDCH. In the simulation it was assumed that DCCH was always present and DCCH rate is 3.4kbps. Rate matching attributes for DTCH and DCCH were assumed to be the same, in other words, the code rate of DTCH and DCCH were the same.

Table F.1: Propagation Channel Models

Case 2, speed 3 km/h	
Relative delay [ns]	**Relative mean power [dB]**
0	0
976	0
20 000	0

Annex G:
Quality evaluation tool

For quality evaluation, an archive *QualityEvaluation_711.zip* is attached to this document. Note that some libraries in the package *ISOFileFormatConverter_711.zip* are required.

The usage of the tool is as follows:

```
QualEval [OrigSeq in 3G format] [ReconSeq in 3G format] [ReceivedSeq in 3G Format]
```

whereby

- OrigSeq corresponds to one the original sequence in 3G format.
- ReconSeq corresponds to the reconstructed sequence without transmission.
- ReceivedSeq corresponds to received sequence after transmission.

The output of the program are the following six values:

- The number of video frames in OrigSeq.
- The number of video frames in ReconSeq.
- The number of video frames in ReceivedSeq.
- The average PSNR in dB.
- The PSNR of the average NSD in dB.
- The PDVD value.

Annex H:
Video test sequences

Test Sequences are contained in the attached Archive *TestSequences.zip*.

Table H.1 contains the test sequence related information. These parameters are also embedded in the ISO-file headers and can be extracted from there as well.

Table H.1: Test sequence related information

No.	File name	Dimensions (pixels)	Frame rate (fps)	Number of frames
1	stunt_qcif.3gp	176x144 (QCIF)	15	240
2	bar-30s.3gp	176x144 (QCIF)	12	480
3	lt-party-30s.3gp	176x144 (QCIF)	12	360

The sequence 'stunt' was donated by Nokia, the capturing was done with a Nokia cell phone. No usage restrictions apply for this sequence.

The sequences 'bar' and 'lt-party' were digitized and donated by the Internet Streaming Media Alliance (http://www.isma.tv/) and may be used for the purposes of this document. For other uses, please contact ISMA.

Annex I:
Encoder-decoder performance verification

To facilitate verification of appropriate usage of encoder, decoder, and quality evalution tool without the usage of the channel, the setup as shown in figure I.1 can be used.

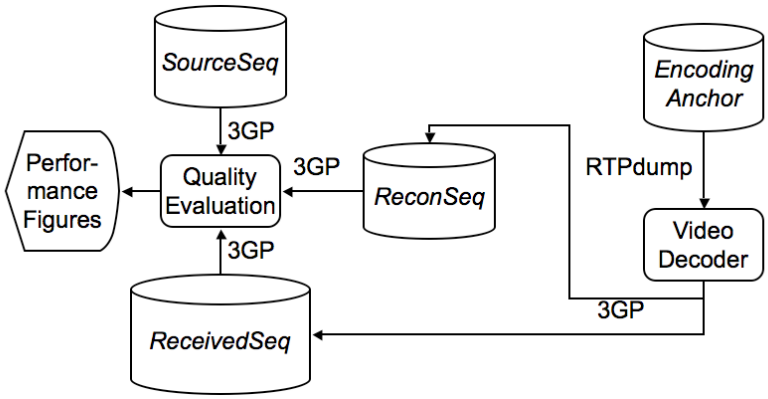

Figure I.1: Encoder-Decoder Performance Verification Setup

The video performance for the select encoding anchors is shown in table I.1.

Table I.1: Selected Video Codec Performance Figures Encoder-Decoder Performance Verification

Test Case	APSNR	PANSD	PDVD
AA1-263	**27.34 dB**	**27.00 dB**	**0**
AA2-263	**30.46 dB**	**29.90 dB**	**0**
AA3-264	**30.79 dB**	**30.71 dB**	**0**
AA4-264	**34.34 dB**	**34.26 dB**	**0**

Note that the verification setup in figure 6 must provide PDVD=0.

Annex J:
Change history

Change history							
Date	**TSG #**	**TSG Doc.**	**CR**	**Rev**	**Subject/Comment**	**Old**	**New**
2007-06	36	SP-070313			Approved at SA#36		7.0.0
2007-12	38	SP-070766	0001	3	Various Corrections to Video Codec Performance	7.0.0	7.1.0
2007-12					editorial clean up	7.1.0	7.1.1
2008-01					correction to history table	7.1.1	7.1.2
2008-12	42				Version for Release 8	7.1.2	8.0.0
2009-12	46				Version for Release 9	8.0.0	9.0.0
2011-03	51				Version for Release 10	9.0.0	10.0.0
2012-09	57				Version for Release 11	10.0.0	11.0.0

History

Document history		
V11.0.0	October 2012	Publication