ETSI TS 102 591 V1.2.1 (2009-06)

Technical Specification

Digital Video Broadcasting (DVB); IP Datacast over DVB-H: Content Delivery Protocols (CDP) Implementation Guidelines

European Broadcasting Union

Union Européenne de Radio-Télévision

EBU·UER

Reference

RTS/JTC-DVB-257

Keywords

broadcast, DVB, DVB-H

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

Important notice

Individual copies of the present document can be downloaded from:
http://www.etsi.org

The present document may be made available in more than one electronic version or in print. In any case of existing or perceived difference in contents between such versions, the reference version is the Portable Document Format (PDF). In case of dispute, the reference shall be the printing on ETSI printers of the PDF version kept on a specific network drive within ETSI Secretariat.

Users of the present document should be aware that the document may be subject to revision or change of status. Information on the current status of this and other ETSI documents is available at
http://portal.etsi.org/tb/status/status.asp

If you find errors in the present document, please send your comment to one of the following services:
http://portal.etsi.org/chaircor/ETSI_support.asp

Copyright Notification

Contents

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://webapp.etsi.org/IPR/home.asp).

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 Specification (TS) has been produced by Joint Technical Committee (JTC) Broadcast of the European Broadcasting Union (EBU), Comité Européen de Normalisation ELECtrotechnique (CENELEC) and the European Telecommunications Standards Institute (ETSI).

NOTE: The EBU/ETSI JTC Broadcast was established in 1990 to co-ordinate the drafting of standards in the specific field of broadcasting and related fields. Since 1995 the JTC Broadcast became a tripartite body by including in the Memorandum of Understanding also CENELEC, which is responsible for the standardization of radio and television receivers. The EBU is a professional association of broadcasting organizations whose work includes the co-ordination of its members' activities in the technical, legal, programme-making and programme-exchange domains. The EBU has active members in about 60 countries in the European broadcasting area; its headquarters is in Geneva.

European Broadcasting Union
CH-1218 GRAND SACONNEX (Geneva)
Switzerland
Tel: +41 22 717 21 11
Fax: +41 22 717 24 81

Founded in September 1993, the DVB Project is a market-led consortium of public and private sector organizations in the television industry. Its aim is to establish the framework for the introduction of MPEG-2 based digital television services. Now comprising over 200 organizations from more than 25 countries around the world, DVB fosters market-led systems, which meet the real needs, and economic circumstances, of the consumer electronics and the broadcast industry.

Introduction

IP Datacast over DVB-H is an end-to-end broadcast system for delivery of any types of digital content and services using IP-based mechanisms optimized for devices with limitations on computational resources and battery. An inherent part of the IP Datacast system is that it comprises a unidirectional DVB broadcast path that may be combined with a bi-directional mobile/cellular interactivity path. IP Datacast is thus a platform that can be used for enabling the convergence of services from broadcast/media and telecommunications domains (e.g. mobile/cellular).

1 Scope

The present document gives implementation guidelines on the use of content delivery protocols and different reliability control techniques in IP Datacast over DVB-H system [1] to [6] and [i.1] to [i.2] for the whole delivery chain from the network to the terminal.

The present document intends in particular to guide implementers of basic use cases for IP Datacast over DVB-H to make best use of the specification IP Datacast over DVB-H: Content Delivery Protocols [2].

2 References

References are either specific (identified by date of publication and/or edition number or version number) or non-specific.

- For a specific reference, subsequent revisions do not apply.
- Non-specific reference may be made only to a complete document or a part thereof and only in the following cases:
  - if it is accepted that it will be possible to use all future changes of the referenced document for the purposes of the referring document;
  - for informative references.

Referenced documents which are not found to be publicly available in the expected location might be found at http://docbox.etsi.org/Reference.

NOTE: While any hyperlinks included in this clause were valid at the time of publication ETSI cannot guarantee their long term validity.

2.1 Normative references

The following referenced documents are indispensable for the application of the present document. For dated references, only the edition cited applies. For non-specific references, the latest edition of the referenced document (including any amendments) applies.

[1] ETSI TS 102 468: "Digital Video Broadcasting (DVB); IP Datacast over DVB-H: Set of Specifications for Phase 1".

[2] ETSI TS 102 472: "Digital Video Broadcasting (DVB); IP Datacast over DVB-H: Content Delivery Protocols".

[3] ETSI TS 102 471: "Digital Video Broadcasting (DVB); IP Datacast over DVB-H: Electronic Service Guide (ESG)".

[4] ETSI TS 102 474: " Digital Video Broadcasting (DVB); IP Datacast over DVB-H: Service Purchase and Protection".

[5] ETSI TS 102 470: "Digital Video Broadcasting (DVB); IP Datacast over DVB-H: Program Specific Information (PSI)/Service Information (SI)".

[6] ETSI TS 102 005: "Digital Video Broadcasting (DVB); Specification for the use of Video and Audio Coding in DVB services delivered directly over IP protocols".

[7] IETF RFC 2327: "SDP: Session Description Protocol".

[8] IETF RFC 3066: "Tags for the identification of languages".

[9] ETSI EN 300 468: "Digital Video Broadcasting (DVB); Specification for Service Information (SI) in DVB systems".

[10] IETF RFC 2616: "Hypertext Transfer Protocol - HTTP/1.1".

[11] IETF RFC 3926: "FLUTE - File Delivery over Unidirectional Transport".

[12] IETF RFC 3451: "Layered Coding Transport (LCT) Building Block".

[13] Void.

[14] IETF RFC 3984: "RTP payload for transport of H.264".

[15] IETF RFC 4425: "RTP Payload Format for Video Codec 1 (VC-1)".

[16] IETF RFC 3640: "RTP payload for transport of generic MPEG-4 Elementary Streams".

[17] IETF RFC 4352: "RTP Payload Format for Extended Adaptive Multi-Rate Wideband (AMR-WB+) Audio Codec".

[18] Void.

[19] IANA: "MIME Media Types".

NOTE: See at http://www.iana.org/assignments/media-types

2.2 Informative references

[i.1] ETSI TR 102 473: " Digital Video Broadcasting (DVB); IP Datacast over DVB-H: Use Cases and Services".

[i.2] ETSI TR 102 469: " Digital Video Broadcasting (DVB); IP Datacast over DVB-H: Architecture".

[i.3] ETSI TR 102 592: " Digital Video Broadcasting (DVB); Guidelines for the implementation of IPDC over DVB-H: Electronic Service Guide (ESG)".

[i.4] ETSI TR 102 377: "Digital Video Broadcasting (DVB); DVB-H Implementation guidelines".

[i.5] ETSI TS 126 234: "Universal Mobile Telecommunications System (UMTS); Transparent end-to-end Packet-switched Streaming Service (PSS); Protocols and codecs (3GPP TS 26.234 Release 7)".

[i.6] ETSI TS 126 234: "Universal Mobile Telecommunications System (UMTS); LTE; Transparent end-to-end Packet-switched Streaming Service (PSS); Protocols and codecs (3GPP TS 26.234 Release 8)".

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the following terms and definitions apply:

associated delivery procedures: set of procedures for file repair and reception reporting which are associated to a file delivery session or a streaming session

base FLUTE channel: first channel signalled in the session description file of file delivery sessions

blocking algorithm: algorithm to chop a file into source blocks and encoding symbols for transport over FLUTE

data download: process of transmitting files from the head-end to the terminals by means of file delivery session

DVB-H: transmission system targeted to provide IP-based services to handheld terminals over terrestrial radio channels, as defined in "Transmission System for Handheld Terminals (DVB-H)"

elementary stream: stream of transport packets within a transport stream sharing a common Packet IDentifier (PID)

NOTE: The "elementary stream" definition differs from the MPEG-2 one.

encoding symbol: array of data bytes that builds up an ALC/LCT packet of a given file

FDT instance: XML document identified by a unique Instance ID that represents a subset of the file data table delivered during the file delivery session

file delivery service: set of files delivered by the server to the terminals in a time-constrained or unconstrained manner

file delivery session: instance of delivery of a file delivery service which is characterized by a start and end time and addresses of the Transport flows used for the delivery of the files between the start and end time

FLUTE channel: as defined in Flute specification RFC 3926 [11] a FLUTE channel is defined by the combination of a sender and destination IP address and port number

NOTE: A receiver joins a channel to start receiving the data packets sent to the channel by the sender, and a receiver leaves a channel to stop receiving data packets from the channel.

IP datacast: end-to-end broadcast system for delivery of any types of digital content and services using IP-based mechanisms

NOTE: An inherent part of the IPDC system is that it comprises a unidirectional DVB broadcast path and a bi-directional mobile/cellular interactivity path.

IP flow: flow of IP datagrams each sharing the same IP source and destination address

LCT channel: as defined in RFC 3451 [12], an LCT channel is defined by the combination of a sender and destination IP address ands port number

post-repair mechanism: set of functionalities supplied by the server and used by the terminals after end of file delivery to recover from unsuccessful reception

NOTE: These functionalities can be based on point-to-point or point-to-multipoint recovery.

reception reporting mechanism: mechanism that defines a request/response procedure for the server and terminals to request and send reception reports

NOTE: Reception reports describe the status of the reception.

source block: set of encoding symbols which is used as the basis for FEC encoding/decoding operations

streaming delivery session: instance of delivery of a streaming service which is characterized by a start and end time and addresses of the Transport flows used for delivery of the media streams between start and end time

streaming service: set of synchronized media streams delivered in a time-constrained or unconstrained manner for immediate consumption (during the reception)

sub-blocking: process of partitioning a source block into small sub-blocks that can be decoded easier with the receivers

NOTE: Sub-blocking may be done at the transmitter or receiver side.

sub-symbol: number of consecutive bytes from a source block of the file

symbol: unit of data processed by the AL-FEC code

NOTE: When sub-blocking is not used, a symbol is a number of consecutive bytes from a source block of the file. When sub-blocking is used a symbol is the concatenation of one sub-symbol from each sub-block of the source block

time slice: burst of MPE and MPE-FEC sections delivered over DVB-H using a time slicing method

transport flow: flow of IP datagrams identified by source IP-address, destination IP-address (either multicast or unicast), port and protocol in use

NOTE: IP flow is composed of one or more transport flows.

transport object: set of source blocks and potentially FEC blocks that build up a given file and which are transported during a file delivery session

3.2 Abbreviations

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

3GPP	3rd Generation Partnership Project
AL-FEC	Application Layer-FEC
ALC	Asynchronous Layered Coding
AS	Application Specific maximum bandwidth
AVC	Advanced Video Coding
AVP	Audio Video Profile
C/I	Carrier to Interference ratio
CDB	Coded Data Buffer
CDP	Content Delivery Protocol
CPB	Coded Picture Buffer
DVB	Digital Video Broadcasting
DVB-H	Digital Video Broadcast-Handheld
ES	Elementary Stream
ESG	Electronic Service Guide
ESI	Encoding Symbol ID
FDT	File Delivery Table
FEC	Forward Error Correction
FLUTE	File deLivery over Unidirectional Transport
HRD	Hypothetical Reference Decoder
HTTP	HyperText Transfer Protocol
IDR	Integrated Decoder Receiver
IP	Internet Protocol
IPDC	IP DataCast
LCT	Layered Coding Transport
MDB	Multiprotocol Decapsulation Buffer
MIME	Multipurpose Internet Mail Extensions
MPE	MultiProtocol Encapsulation
MPEG-2	Motion Pictures Experts Group session 2 transport stream
NTP	Network Time Protocol
PSI/SI	Program Specific Information/Service Information
RDB	RTP Decapsulation Buffer
RFC	Request For Comments
RR	bandwidth modifier for RTCP Reception Reports
RS	bandwidth modifier for RTCP Sender Reports
RTCP	Real-time Transport Control Protocol
RTP	Real-time Transport Protocol
SBN	Source Block Number
SDP	Session Description Protocol
SNTP	Simple Network Time Protocol
TCP	Transmission Control Protocol
TDT	Time and Date Table
TIAS	Transport Independent Application Specific maximum bandwith
TOI	Transport Object Identifier
TOT	Time Offset Table
TS	Transport Stream
TSI	Transport Session Identifier
UDP	User Datagram Protocol
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
UTC	Universal Time, Co-ordinated
UTF	Unicode Transformation Format
XML	eXtensible Markup Language

4 System overview

IP Datacast over DVB-H is an end-to-end broadcast system for delivery of any types of digital content and services using IP-based mechanisms, optimized for devices with limited computational resources and battery power. The IPDC system is designed to carry two different types of delivery methods: streaming delivery and file delivery. Streaming delivery methods are used to realize e.g. mobile TV and radio services. File delivery methods are used to carry, e.g. the Electronic Service Guide and for the download of audio-visual content and software applications.

Figure 1 gives an overview of the end-to-end system functions of the IPDC system.

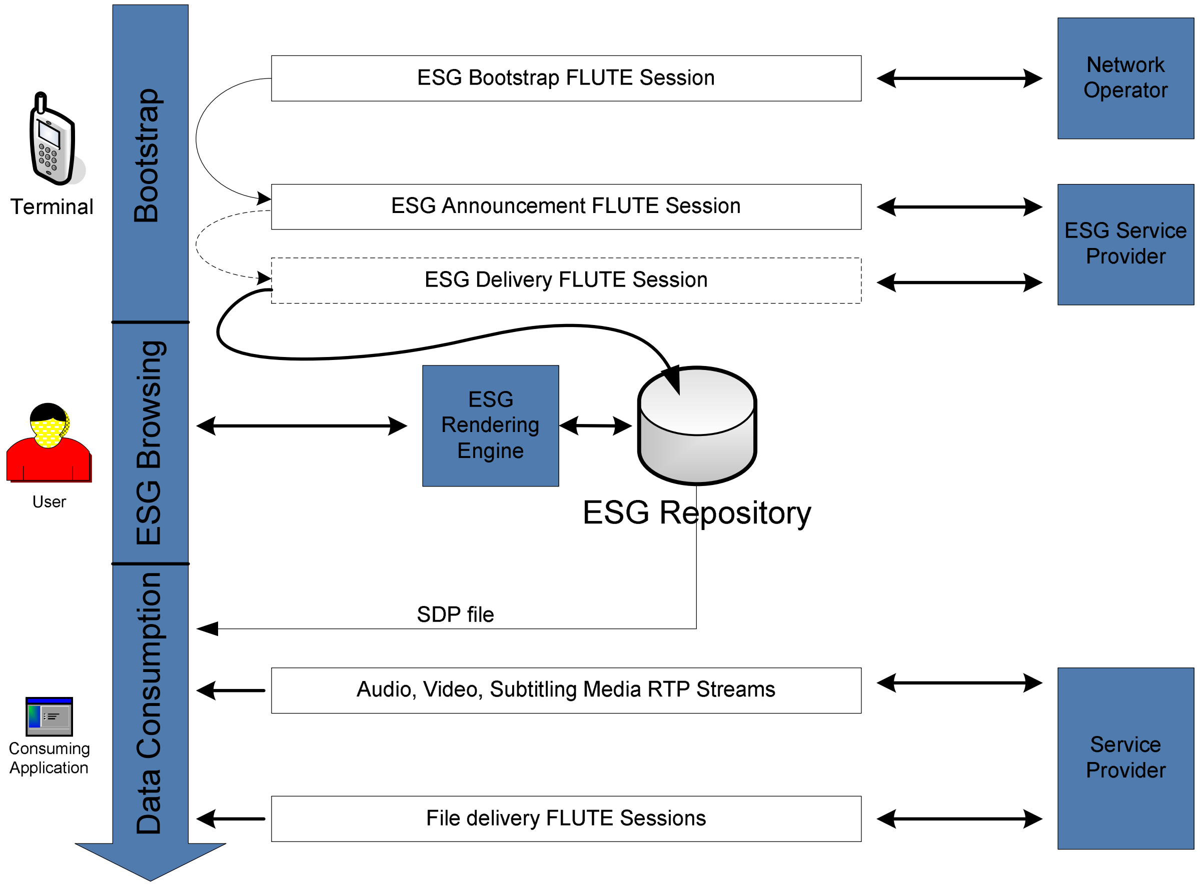

Figure 1: IPDC end-to-end system functions

Figure 1 describes the typical procedure of receiving and consuming an IPDC service starting from the ESG bootstrap session. The service delivery methods are based on standard protocols and media formats as well as extension mechanisms for service specific signalling. RTP is used for media streaming, FLUTE is used for file delivery, SDP is used for session description, and XML data is used for signalling different functionalities of the service (e.g. file repair in file delivery sessions).

In this guidelines document, the implementation of different use cases based on the specified delivery methods and in accordance with the Content Delivery Protocols TS 102 472 [2] and Electronic Service Guide TS 102 471 [3] specifications is discussed.

5 Mobile TV and radio services

This clause gives guidelines to setup and operate a streaming session for Mobile TV and Radio Services. The clause also gives examples of implementing the hypothetical buffering model. Examples of signalling and delivering of Subtitling and Dynamic zapping service are also covered.

5.1 Setup and operation of a streaming session

This clause describes the initialization and running of streaming services.

5.1.1 Session initiation using the ESG

This clause describes the session initiation starting from the ESG with the necessary parameters that are extracted from the ESG and the session setup procedure.

5.1.1.1 Initiation of an audiovisual session

In this clause, the parameters that are required to initiate an RTP session are provided. In order to initiate an RTP session the terminal must know:

- The parameters of the session.
- The media codecs in use.
- The timeframe of the session.
- The IPPlatformID in use, in order to perform IP address resolution at PSI/SI level. This information is usually available from the bootstrap phase of the DVB-H device, where it selects the IP Platform over which it will operate. The IP Platform to be used for reception of services is the same as that of the ESG.

5.1.1.1.1 Streaming session parameters

The parameters of a session are provided in the Session Description of the ESG Acquisition fragment that either:

- directly describes a session by:
  - an inlined SDP file of the ESG Acquisition fragment (case 1).
- or indirectly describes a session by:
  - pointing to an SDP encapsulated in an ESG container (case 2);
  - pointing to an SDP file transported within an ESG FLUTE session (case 3);
  - pointing to an SDP transported in a separate FLUTE session that is announced by an inlined SDP (case 4).

The different cases are illustrated in figure 2.

In indirect referencing (cases 2 to 4), the reference (SessionDescription) consists of:

- an SDP description (SDPStream) that describes the FLUTE session that carries the SDP file of the service, and
- the URI of that SDP file (SDPURI).

Based on this information the terminal is able to join the referenced FLUTE session. The terminal may already be tuned to the FLUTE session that includes the targeted SDP (cases 2a and 3a). The referenced FLUTE session may be another ESG delivery session (cases 2b and 3b) or out-of-band FLUTE session (case 4).

The terminal may discover the targeted SDP file from the FDT of the FLUTE session, i.e. SDPURI matches FDT Content-Location (cases 3 and 4).

Alternatively, the terminal may need to search the ESG containers for an ESG fragment that corresponds to the SDP file of the session (case 2). This is done by matching the SDPURI to the metadataURI of the ESG auxiliary data. The mimetype of the ESG auxiliary data has to be "application/sdp".

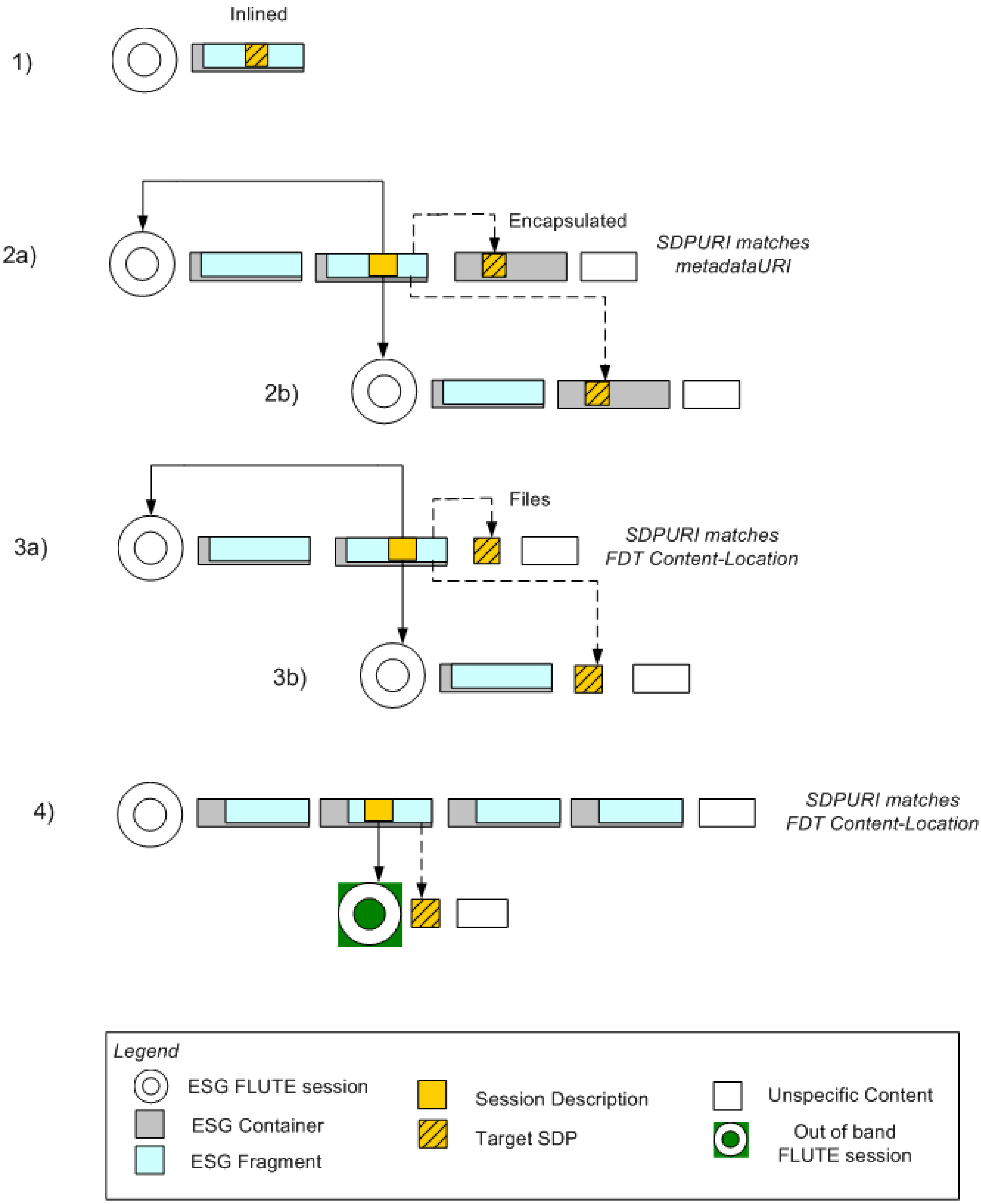

Figure 2: Cases to retrieve and deliver SDP describing delivery sessions

Depending on the type of the SDP delivery, the terminal may detect updates of the SDP as follows:

- Inlined SDP (case 1): an update of the SDP is detected by watching for updates to the ESG containers and consequently acquisition fragment that carries the SDP.
- SDP as ESG auxiliary data (case 2): the terminal locates the SDP and registers its fragment_id and version. To detect updates to the SDP, the terminal checks the ESG Fragment Management Information for announcement of a new version of the fragment (identified through its fragment_id).
- SDP file is carried as a single file in the FLUTE session (cases 3 and 4): the terminal checks the FDT to detect the announcement of a new version of the file, which is done by announcing a new mapping between the file URI and the TOI in a new FDT instance. More guidelines on the terminal behaviour in case of dynamic session updates are given in clause 5.1.2.

The session description includes the following information:

Network layer information:

- Source IP address(es) from the source-filter attribute. In case this attribute is not present and the information is not available from other means (e.g. out of band), then the terminal can only filter the streams from the IP destination address. It means that the terminal has no means to filter out streams from the trusted source(s) only.
- Destination IP address from the "c=" field.

Transport layer information:

- RTP destination port from the port subfield in the "m=" line.

NOTE: Specification TS 102 472 [2] in clause 5.2.1.2 mandates RTCP port to be RTP port +1.

5.1.1.1.2 Media codec information

The terminal retrieves media codec information from the Session Description as specified in RFC 2327 [7] for the RTP Audio/Video profile. Additionally, the ESG provides media information in the ComponentCharacteristics part of the Acquisition fragment that may be used by the media player.

5.1.1.1.3 Timing parameters

The timing parameters include the start and end time of the session. This information is provided in the ScheduleEvent fragment with the PublishedStartTime and PublishedEndTime elements and in the "t=" line in the SDP.

The user should only see the timing published in the ESG and select content based on this timing. The media player is expected to consume the sdp delivered along with the ESG or out-of-band. It is not expected that the terminal will modify timing information contained in the SDP file based on the ESG information, therefore the sdp should have some relaxed session timing boundaries (i.e. larger than the timing boundaries in the ESG). Live streaming session with SDP session timing boundaries "t=0 0" should be considered as a permanent session.

In case of dynamic session updates, see clause 5.1.2.

5.1.1.2 Initiation of multi-lingual sessions

When one or more audio tracks representing different languages are part of a multimedia stream that also provides video tracks, the terminal must be able to map the language selected by the user with the corresponding audio tracks.

The signalling of the audio media streams is done in the session description (SDP) and can also be available in the ESG. In the ESG, the Acquisition fragment provides information about an available audio stream with the ComponentCharacteristics element of type AudioComponentType. Such element holds a Language element that indicates the language of the audio stream. In the SDP, the "lang" attribute is used to identify the language of each of the audio streams in the media description section. Several audio streams with different languages may be declared in the Acquisition fragment as well as in the session description, along with the video stream.

Within IPDC, languages are represented in SDP according to the rules defined in RFC 3066 [8]. It is recommended that the values of the "Language" element follow the same rules so that language signalling remains consistent across SDP and ESG.

An example instantiation of a content with one video stream and two audio streams is provided below. In this example, tracks in English and French are available.

```
<?xml version="1.0" encoding="UTF-8"?>
<ESGMain xmlns="urn:dvb:ipdc:esg:2005" xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001"
xmlns:tva="urn:tva:metadata:2005" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >
   <ESG>
   <ContentTable>
      <Content contentID="cbms://foo.com/Content1">
         <Title xml:lang="en">Content Item 1</Title>
         <Duration>PT30M</Duration>
      </Content>
   </ContentTable>

   <ScheduleEventTable>
      <ScheduleEvent>
         <PublishedStartTime>2006-03-02T21:30:00</PublishedStartTime>
         <PublishedEndTime>2006-03-02T23:30:00</PublishedEndTime>
         <ServiceRef IDRef="cbms://foo.com/Channel1" />
         <ContentFragmentRef IDRef="cbms://foo.com/Content1" />
      </ScheduleEvent>
   </ScheduleEventTable>

   <ServiceTable>
      <Service serviceID="cbms://foo.com/Channel1">
         <ServiceName>Channel1</ServiceName>
         <AcquisitionRef IDRef="cbms://foo.com/Acquisition/Channel1" />
      </Service>
   </ServiceTable>

   <AcquisitionTable>
      <Acquisition contentMimeType="video/H264" acquisitionID="cbms://foo.com/Acquisition/Channel1"
>
         <ComponentDescription>
            <ComponentCharacteristic xsi:type="VideoComponentType">
               <CodecCharacteristic>
                  <Codec href="urn:dvb:CS:Video:4.5.6"/>
               </CodecCharacteristic>
               <FrameRate>25</FrameRate>
            </ComponentCharacteristic>
            <ComponentCharacteristic xsi:type="AudioComponentType">
               <Codec href="urn:dvb:CS:Audio:4.2"/>
               <Language>en</Language>
            </ComponentCharacteristic>
            <ComponentCharacteristic xsi:type="AudioComponentType">
               <Codec href="urn:dvb:CS:Audio:4.2"/>
               <Language>fr</Language>
            </ComponentCharacteristic>
            <SessionDescription xsi:type="SDPRefType" >
               <SDPStream>
                  <![CDATA[v=0
o=foo.com 751092616 751111042 IN IP6 FE80::1:4D3E
s=SDP Delivery for Channel1
t=3350323000 0
a=flute-tsi:9532
a=flute-ch:1
a=source-filter: incl IN IP6 * FE80::2::70CA
m=application 12345 FLUTE/UDP 0
c=IN IP6 FF15::1:141B
]]>
                  </SDPStream>
               <SDPURI>urn:dvb:ipdc:esg:sdp</SDPURI>
            </SessionDescription>
         </ComponentDescription>
      </Acquisition>
   </AcquisitionTable>
</ESG>
</ESGMain>
```

Again, note that the instantiation above gives one level of indirection to the SDP file that is needed by the terminal to consume the multimedia session. Below is an example of such an SDP file (fields that map to the ESG instantiation are in bold):

```
v=0
o=IPDC 2890844526 2890842807 IN IP6 FE80::5F:E47D
s=IPDC SDP language example
i=An example of session description for a multimedia content with two languages
c=IN IP6 FF1E:03AD::7F2E:172A:1E24
b=AS:77
t=3350323800 3350331000
a=source-filter: incl IN IP6 * 2001:210:1:2:240:96FF:FE25:8EC9
a=min-buffer-time:500
m=video 4002 RTP/AVP 96
c=IN IP6 FF1E:03AD::7F2E:172A:1E24
b=TIAS:62000
b=RR:0
b=RS:600
a=maxprate:17
a=avg-br:48000
a=framerate:25
a=rtpmap:96 H264/90000
a=fmtp:96 profile-level-id=42A01E; packetization-mode=1; sprop-parameter-sets=Z0IACpZTBYmI,aMljiA==
m=audio 4004 RTP/AVP 98
c= IN IP6 FF1E:03AD::7F2E:172A:1E25
b=TIAS:15120
b=RR:0
b=RS:600
a=maxprate:10
a=avg-br:14000
a=rtpmap:98 AMR/8000
a=fmtp:98 octet-align=1
a=lang:en
m=audio 4006 RTP/AVP 100
c= IN IP6 FF1E:03AD::7F2E:172A:1E26
b=TIAS:15120
b=RR:0
b=RS:600
a=maxprate:10
a=avg-br:14000
a=rtpmap:100 AMR/8000
a=fmtp:100 octet-align=1
a=lang:fr
```

Note that the SDP examples use IPv6 addresses throughout. However, IPv4 addresses can be used as well. Mixing IPv4 and IPv6 addresses in an SDP file is discouraged. An exception to this recommendation is the IP address in the origin field, which may be given as an IPv4 address as the session creator of the streams would typically have an IPv4 address.

The destination IP address is recommended to be declared separately for each media stream of the streaming session, i.e. a connection field should appear at the media level for each of the media lines of the SDP. The example above depicts the case where each media stream is sent to a different destination IP address. The connection information at media level overrides the connection information at session level.

5.1.2 Dynamic session update

Dynamic session update, as described in the present document, is the operation by which the sdp file of an on-going session is replaced by a newer one, signalled through appropriate means.

In these guidelines, only the case of single sdp file update is addressed.

The ESG data is typically delivered in a different Elementary Stream (ES) and possibly also in a different Transport Stream than the file delivery and streaming sessions it describes. In such a case, a terminal typically updates its ESG database upon tuning in and then switches to the selected stream for service consumption. During service consumption, it is assumed that the terminal may miss updates to the ESG as tracking them all could increase the power consumption and may not be possible (e.g. in case of different Transport Streams).

However, some services such as a "TV channel" service might need to update the ESG in an unpredictable manner. As an example, a new audio language or a subtitling stream appears suddenly in the TV channel contents. In another case, an already scheduled event may be postponed due to some delays. In such scenarios, the session description file of the streaming session needs to be updated, in order to enable correct consumption of the service. Furthermore, terminals that are consuming the service and not tuned to the ESG have to be made aware of the changes to the session parameters.

The ESG specification [3] defines for this purpose an out of band delivery mechanism for SDP files. A FLUTE session may be used to deliver the most up to date SDP file to the terminals that are currently consuming or just tuning to the service. The most optimized (power efficient) way is to have the FLUTE session delivered in the same Elementary Stream as the media streams of the streaming service to consume. In this case, the terminal first initiates reception of the corresponding FLUTE session and retrieves the SDP file before being able to launch the media player.

However, in order to allow compatibility with terminals that do not support dynamic session update, a default SDP file has to be provided. This is typically done through the Acquisition fragment that relates to the Service fragment describing the service, in a way similar to the one in clause 6.4 "TV Service with additional options" of [i.3]. In this situation, the terminal can pre-fetch and store the SDP files in advance so that once the user selects a service to consume, the media player can be launched right away.

In the following clauses, different aspects of the session description update are described.

5.1.2.1 Timing synchronization

Terminal consuming a FLUTE session for dynamic session update concurrently to a streaming service, is supposed to download any new version of the SDP file. As described in CDP specification [2], a new version is identified by a different TOI to URI mapping that appears in a newer FDT instance. The URI of the SDP file is obtained from the SDPURI element contained in the SessionDescription element of the Acquisition fragment. This SessionDescription element is of type "SDPRefType" so as to signal that the SDP file is not inlined in the Acquisition fragment but delivered in a separate FLUTE session.

Upon detection of a new version of the SDP file of the streaming session, the terminal checks the start time in the SDP "t=" field. The start time is given as NTP timestamp (UTC time). The terminal should use the Sender Reports sent in the RTCP streams of the streaming service to establish the time synchronization and to schedule the update of the session with the new SDP file. This procedure is described in figure 3.

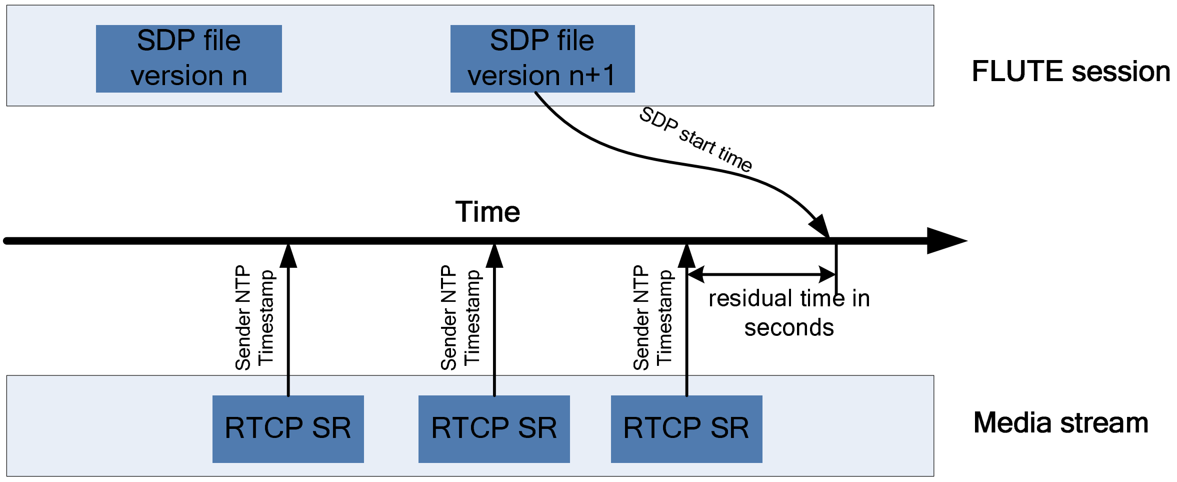

Figure 3: Time Synchronization in dynamic sessions

This provides for a more robust time synchronization as it binds the session updates to the media playout timeline rather than to the wallclock. This would allow for accounting for media buffering and transmission jitter (due to time-slicing).

5.1.2.2 Detection of dynamic session update events

During the lifespan of a service, dynamic session updates may not always be in use. For example, the first instantiation of a service might only involve a Service-related Acquisition fragment, providing a default SDP. In a following instantiation, a ScheduleEvent-related Acquisition fragment is provided, pointing to the FLUTE session related to dynamic session update. This is shown in figure 4a.

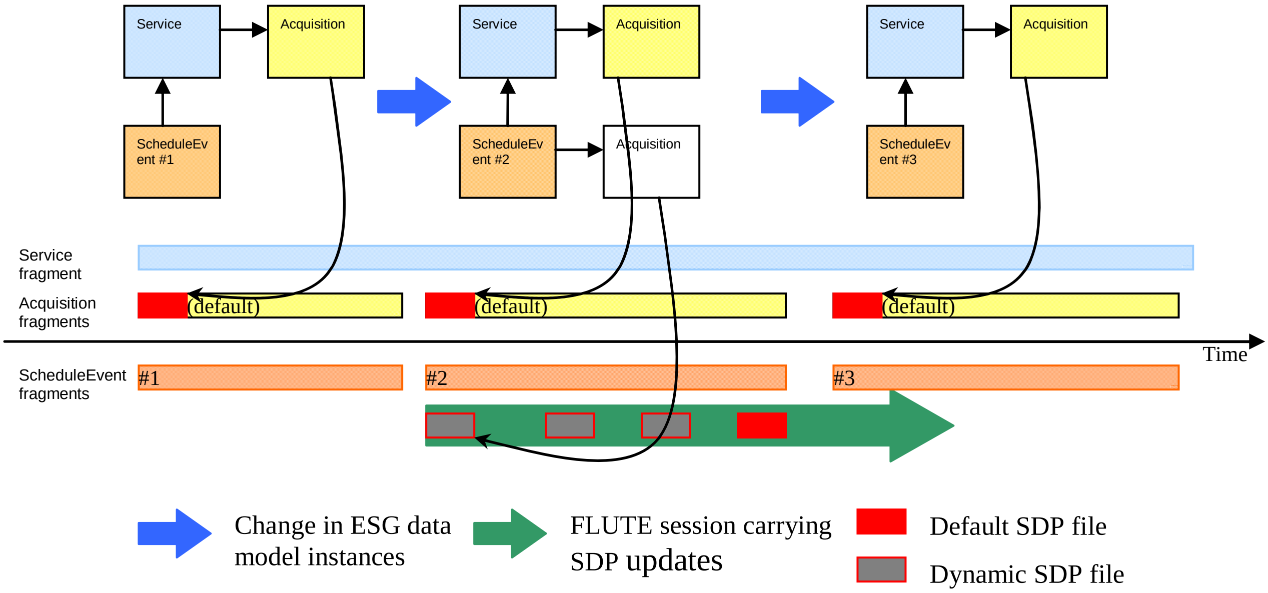

Figure 4a: Evolution of a service and availability of dynamic session updates during a single event.

As a consequence, tuning to the FLUTE session related to dynamic session updates will not provide all the update events for a given service. In order to catch them all, it is necessary that the terminal query the ESG on an appropriate frequency (e.g. prior to each Schedule Event change). The ScheduleEvent fragments hold the PublishedEndTime of the scheduled events. The terminal can use this information to query the ESG before the current ScheduleEvent gets depreciated.

To enable a seamless transition from a dynamic SDP to the default SDP at the end of ScheduleEvent, it is recommended that the current default SDP is transmitted in the FLUTE session as a new version of the dynamic SDP (as shown in figure 4a).

Services that are expected to be highly dynamic should make use of the dynamic session update mechanism throughout their lifetime, provided a default SDP remains available through a Service-related acquisition fragment. It is recommended to have a single FLUTE session that signals the dynamic SDP updates throughout the lifetime of the service. This case is illustrated in figure 4b.

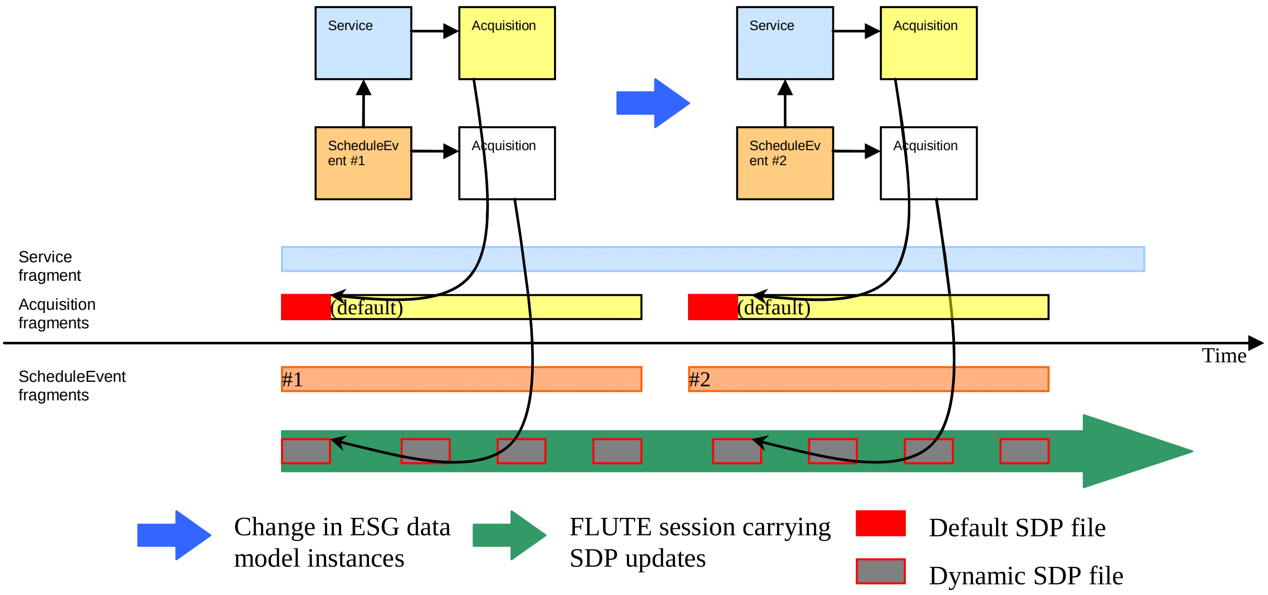

Figure 4b: Evolution of a service and availability of dynamic session updates for highly dynamic services throughout their lifetime.

5.1.2.3 Media player behaviour

Upon reception of a new version of the SDP that relates to the streaming service being consumed, the terminal calculates the activation time of the new SDP file as discussed previously and schedules the update of the session. The terminal should preferably omit complete tear down and re-initialization of the streaming session.

In a first scenario, the terminal analyzes the updated SDP to detect whether any of the currently consumed media streams is being discontinued or updated. A media stream is identified by its destination IP address and port number. In such case, the terminal needs to instruct the media player how to continue consumption of the service. Several options are possible:

- The terminal may stop the reception of the discontinued media component completely. This is e.g. practicable when an auxiliary media stream such as a subtitling stream is discontinued.
- The terminal may replace the discontinued media stream by an equivalent media stream in the updated SDP file. An audio stream can replace another audio stream and a video stream can replace a video stream. The choice of the media stream may be based on user preference (e.g. revert to the user preferred audio stream with a specific language). The terminal may also prompt the user to select the preferred media streams to be consumed from the session. If only one alternative for audio and video streams is available, the media player is instructed to consume the available media stream without further interaction with the user. When the media stream is updated (i.e. some parameters are added to or changed in an existing media declaration), the natural choice for the player is to continue with the updated stream. However, unless the player is able to seamlessly use the new update, the media session is likely to be reset and reloaded.
- The terminal may stop the service consumption completely. This is reasonable if e.g. the session is declared to have ended (by setting the session end time less or equal the session start time).

The last scenario is that the new SDP file declares new media lines (possibly in addition to the ones being currently consumed). In this case, the terminal has several options on how to behave:

- The terminal continues playout of the currently initialized media streams.
- The terminal decides for each media type whether to switch to an alternative media stream or possibly also to consume several media streams of the same type simultaneously (e.g. in case of additional audio channels in a new media stream). This decision may be based on user preference or on user interaction.
- The terminal decides whether to consume a new media stream of a new media type (e.g. a new subtitling stream is added) based on pre-configured user preference or on user interaction.

In order to ensure smooth playout, the media player can buffer the requested amount of data of the new alternative media stream before interrupting the playout of the original media stream. The media player can also perform the switch to the new media stream at a media sample as close as possible to the last media sample played out from the old media stream.

5.1.3 Implementation of the hypothetical buffering model

In this clause, an example of implementing the hypothetical buffering model is provided. The simulation setup is first described and then operation of the hypothetical buffering model is analysed.

5.1.3.1 Simulation setup

The simulations are based on the parameter selection scenarios described in annex C of the DVB-H implementation guidelines TR 102 377 [i.4]. The following parameters were used to setup the simulation environment for the simulations:

- Time-sliced mode is simulated.
- Encoder: Nokia open-source H.264 codec is used as video encoder. A rate control algorithm is used to produce an HRD compliant stream with the following parameters as input:
  - average stream bitrate;
  - IDR picture frequency;
  - Virtual Buffer Size;
  - Initial Buffering Period is adjusted by the rate control to:
    - 0,6 × Virtual Buffer Size / Target Bitrate.
- Multiplexer:
  - The number of bits of each elementary stream that is available for multiplexing is variable for each data burst. This can result from delay jitter in the back-end during the delivery of stream from the service provider to the multiplexer. We simulate this variation using a simple sinusoidal probability function that assures an average input rate equal to the stream average bitrate.
- IP Encapsulator:
  - Burst size.
  - ES (Elementary Stream) peak bit rate.
  - No. of services per ES.
  - Cycle Time.
  - ES Average Bitrate.
  - MPE-FEC Code Rate.

The other required parameters are computed based on these inputs:

- Channel: A transmission channel with a uniform distribution of the TS packet error rate is assumed for simplicity.
- MDB: Multi-Protocol Decapsulation Buffer is simulated according to the CDP specification.
- RDB: The value of the average bitrate of the stream and the Initial Buffering Delay "a=min-buffer-time" are used as controlling parameters for the simulations.
- CDB: For video, the HRD CPB (Coded Picture Buffer) as specified in H.264/AVC was simulated. The input of the CPB is simulated according to output of the RTP Decapsulation Buffer.

- Video Content: A video sequence with 10 000 frames and a large amount of motion and scene cuts was used for the simulations.
- Audio Parameters: The considered service consists of one video and one audio stream. the following parameter are used for the audio stream as input to simulator:
  - Bitrate.
  - Frame Size.
  - Number of Frames Per Packet.
- Other Simulation Parameters: Simulator uses other internal parameters which can be tuned. Time scale is the most important parameter as it controls the accuracy of the simulation results. A large time scale increases the accuracy of the simulation results while it needs a long time to run. On the other hand a small time scale may provide inaccurate results.

5.1.3.2 Simulations and results

Simulations were performed with the following parameter configurations.

IP Encapsulator Parameters:

- Maximum Burst size = 1 Mbits.
- ES (Elementary Stream) peak bit rate = 5 000 000 bits/s.
- No. of services per ES = 1 Video + 1 Audio (using 600 000 bits, the rest is used for other services).
- Cycle Time = 2 seconds.
- MPE-FEC Code Rate = 3/4.

Encoder Parameters:

- Bitrate = 128 000 bits/s.
- Frame rate = 15 Frames/s.
- IDR frequency = 1 every 2 seconds.
- Virtual Buffer Size = 0,8 seconds × Bitrate.
- Initial Buffering Period = 0,6 × Virtual Buffer Size / Average Bitrate.
- Max Video Slice Size = 1 200 Bytes.

Audio Parameters:

- Bitrate = 64 000 bits/s.
- Frame Size = 100 Bytes.
- No. of Frames Per Packet = 8 Frames.

The simulation results are as follows:

Minimum buffering delays are:

- RDB minimum buffer delay = 0,32 s.
- Coded Data Buffer: 0,48 s.
- HRD Coded Picture Buffer: 0,48 s.

Minimum buffer sizes:

- RTP Decapsulation Buffer: 296 604 Bits.
- Coded Data Buffer: 105 810 Bits.
- HRD Coded Picture Buffer: 97 648 Bits.

Figure 5 depicts the results for the hypothetical receiver buffer model. The time scale used for the simulations is 1 to 300, i.e. 1 tick corresponds to 1/300 s. The fullness of CDB and HRD CPB are very similar throughout the simulation duration.

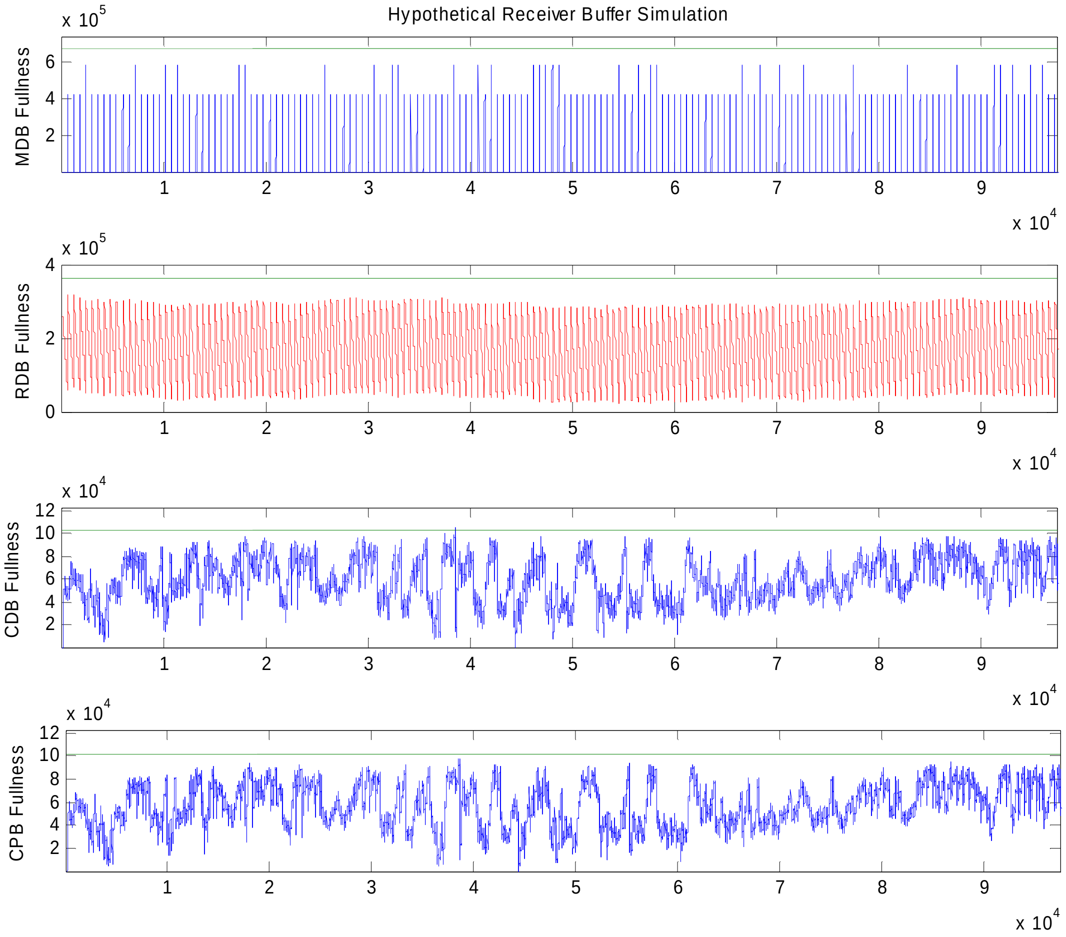

Figure 5: Buffer fullness of the MDB: Multiprotocol Decapsulation Buffer, CDB: Coded Data Buffer, CPB: Coded Picture Buffer according to the HRD model

More simulations have been performed to gather results for "class B" streams with the following encoding parameters:

Video Parameters:

- Bitrate = 300 000 bits/s.
- Frame rate = 15 Frames/s.
- Picture Format = QVGA (320×240).
- IDR frequency = 1 every 2 seconds.
- Virtual Buffer Size = 0,8 seconds × Bitrate.
- Initial Buffering Period = 0,6 × Virtual Buffer Size / Average Bitrate.

- Max Video Slice Size = 1 000 Bytes.

Audio Parameters:

- Bitrate = 64 000 bits/s.
- Frame Size = 100 Bytes.
- No. of Frames Per Packet = 8 Frames.

Four different video clips (with different characteristics) have been used for performing the simulations, in order to cover a wide range of possible content types.

The results are as follows:

Table 1: The buffer fullness results for the case of the sport video clip

Video Sequence	Initial Buffering Period in Simulation (Second)			Minimum Initial Buffering Period (Second)			Minimum Buffer Size (Kbit)		
	RDB	CDB	CPB		RDB	CDB	CPB		RDB
Sport	0,20	0,48	0,48	Sport	0,20	0,48	0,48	Sport	0,20
News	0,20	0,48	0,48	News	0,20	0,48	0,48	News	0,20
Animation	0,20	0,48	0,48	Animation	0,20	0,48	0,48	Animation	0,20
Music	0,20	0,48	0,48	Music	0,20	0,48	0,48	Music	0,20

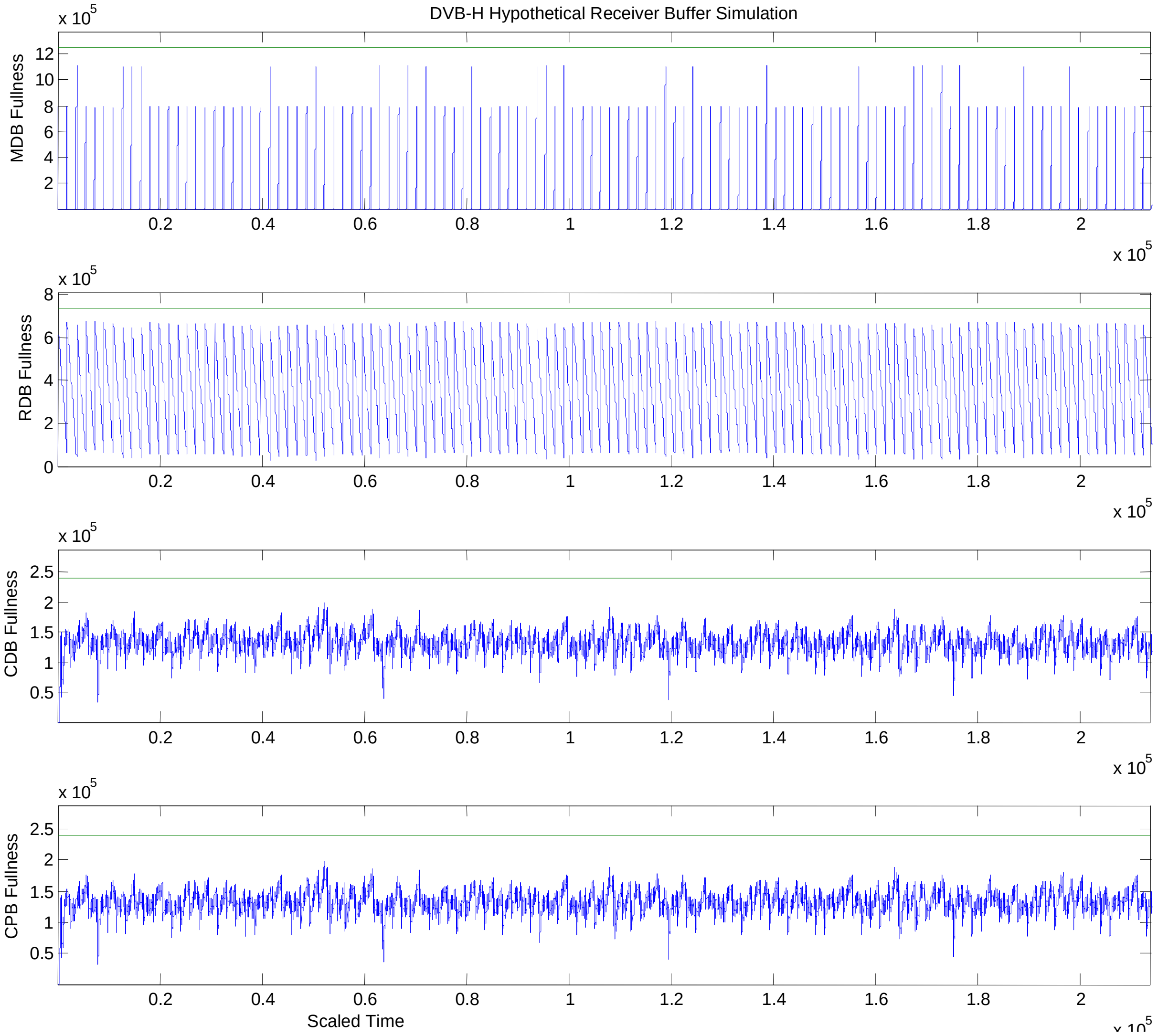

Figure 6: Buffer fullness results for Sport video clip

Again, the buffer fullness of the Coded Data Buffer (CDB) prove to be conformant to the Coded Picture Buffer (CPB), which means that the output of the RTP Decapsulation Buffer (RDB) is conformant to the H.264/AVC HRD.

5.1.3.3 Analysis of the hypothetical buffering model

In the following, we describe the functioning of the hypothetical buffering model.

The output of the media encoder is typically contained within a leaky bucket with a given size B, initial buffer delay $D_{HRD,}$ and the average output bitrate R of the media.

The IP Encapsulator creates the data bursts for the elementary stream carrying the service. The IP Encapsulator is typically the entity of the network that implements the hypothetical receiver buffer model to make sure that the produced elementary stream and the contained media streams can be handled by the terminals that provide the necessary initial buffering.

The hypothetical buffering model consists of two main components, which are the MPE Decapsulation Buffer and the RTP Decapsulation Buffer. We can assume that the MDB does not overflow as long as the time cycle is larger than the MPE-FEC decoding and data processing time by the terminals. The RDB is more problematic as it will provide the input to the CDB (coded data buffer of the media decoder). The RDB converts the bursty data incoming from the DVB-H network into a regular constant bitrate output stream equivalent to the media encoder output. This is necessary to guarantee that HRD conformant media streams are decodeable correctly in the DVB-H environment.

In the following, the behaviour of the RDB is described to give a better understanding of its details. The RDB occupancy should not underflow or overflow at any time instance given an initial buffering time. The initial buffering time has to be appropriate enough to suit any session joining time of the terminal.

Let b^j_i be the RDB buffer occupancy of media stream j at time t_i, Bmax be the maximum RDB buffer size, and s^j_i be the amount of data of stream j packet the data burst sent at time t_i. The RDB is being drained at a rate R equivalent to the media stream average bitrate. The following equation is valid:

$$b_i^j = \max\left(\min\left(b_{i-1}^j + s_i^j - R \times (t_i - t_{i-1}), B_{\max}\right), 0\right) \qquad (1)$$

Now assuming no buffer overflow or underflow, the previous equation is equivalent to:

$$b_i^j = b_{i-1}^j + s_i^j - R \times (t_i - t_{i-1}) \qquad (2)$$

By consequence, assuming that the terminal joins the streaming session and receives the first data burst at time t_s , that no overflow or underflow occurs during the period between t_s and t_i, and that the required minimal buffer time is D, b^j_i is then calculated as follows:

$$b_i^j = \sum_{l=0}^{i} s_l^j - R \times (t_i - t_s - D) \qquad (3)$$

Assuming that the delay jitter is negligible, the time period between the reception of burst s (t_s) and the reception of burst i (t_i) is equal to the time period between transmission of burst s (tt_s) and transmission of burst i (tt_i).

The network has to set the delay D, the time cycle (t_i-t_{i-1}), and the data transmitted in each data burst (s_i^j) for a given media stream in way so that for any pair (s,i), s < i, the following equation holds:

$$0 \le b_l^j \le B_{\max}, \text{for s} \le \text{l} \le \text{i} \qquad (4)$$

This will assure that terminals joining the session at any point of time will be able to correctly process the media streams.

5.1.4 Media encoding, transport, and decoding

The media formats used for media streaming in IPDC over DVB-H are defined in [6]. For the audio and video media formats, the following media codecs are allowed:

Audio:

- MPEG-4 High Efficiency Advanced Audio Coding (HE AAC v2). MPEG-4 AAC and MPEG-4 HE AAC profiles are also permitted.
- Extended Adaptive Multirate Wide-band (AMR-WB+).

Video:

- H.264 AVC Baseline profile at levels 1b, 1.2, and 1.3. The level selection depends on the terminal capability class (A, B or C respectively).
- VC-1 Simple profile at levels LL and ML and Advance profile at level L0. The level selection depends on the terminal capability class (A, B, or C respectively).

H.264/AVC video is encapsulated using RTP payload format for H.264/AVC video [14], but in case ISMACryp is used for content protection, H.264/AVC video is encapsulated using RTP payload format as specified by [4]. When H.264/AVC is encapsulated using RTP as defined by [14], only the single NAL unit and the non-interleaved packetization modes are allowed. As a consequence, after arranging of the RTP packets to RTP sequence number order by the RTP layer,NAL units arrive at their decoding order at the H.264 de-payloadizer. CDP specification defined a recommended packet size of 1 400 bytes in order to avoid packetization at the IP layer. This would allow for RTP payload of more than 1 300 bytes. Depending on the selected terminal capability class (A, B, or C), the resulting picture size may be small enough to fit into the payload of an RTP packet. In such a case, the non-interleaved mode is used to reduce the overall protocol overhead. The non-interleaved mode also covers the single NAL unit mode as it supports encapsulation of a single NAL unit to an RTP packet. The RTP encapsulator should if possible avoid creating FU-A packets for the non-interleaved mode as fragmentation is not desirable. However, some NAL units may exceed the target payload size and might need fragmentation.

To allow for fast tune in and channel switching in a coding efficient manner, [6] recommends the insertion of random access points at least once every 2 seconds. It also recommends that data burst start with a random access point. Specification [6] mandates a random access point to be an IDR access unit. A decoder should be capable of producing correct output pictures no later than when the random access point is presented. The "fmtp" attribute of the SDP may carry the sequence and picture parameter sets of the video sequence, in which case those can be omitted in the random access point.

VC-1 video is encapsulated using RTP payload format for VC-1 video as defined in [15]. That allows for packing multiple access units into a single RTP packet. The "fmtp" attribute of the SDP carries decoder initialization metadata. Updates to those parameters are only allowed in-band for the advanced profile.

HE AAC v2 is encapsulated using RTP payload format defined in [16]. AMR-WB+ is encapsulated using RTP payload format defined in [17]. Both payload formats allow for multiple frame encapsulation in one packet. Frame interleaving is optional.

5.2 Subtitling

This clause gives descriptive examples of subtitling media streams and how they are mapped to a video stream in a streaming session.

5.2.1 Signalling of subtitle streams

CDP defines two different options for realizing a subtitling stream as part of a multimedia stream. The first option is a text-based representation of subtitling data and is defined based on the 3GPP Timed Text format. The second option is a bitmap-based representation as defined in the DVB subtitles, whereas the transport is based on the RTP payload format for the transport of MPEG-2 streams.

To inform user about the availability of multiple closed captions related to a particular content it is recommended to declare the caption languages with the CaptionLanguage element of the Content Fragment in the ESG. A terminal can detect the existence of subtitling streams from the SDP related to that Content Fragment via a ScheduleEvent Fragment or via a Service Fragment. The latter is the case if it is assumed that the subtitles persist for the complete Service.

The signalling of the subtitling media stream is done in the session description (SDP). In the SDP the "a=lang" attribute is used to identify the language of each of the subtitling streams. Several subtitling streams with different languages may be declared in the session description, along with the audio and video streams. Within IPDC, languages are represented in SDP according to the rules defined in [8].

An example instantiation of a content with one or more subtitling streams is provided below. In this example, subtitles in English and French are available for a content which is in English. Here the SDP describing the subtitling streams relates to the Content Fragment via a ScheduleEvent Fragment.

```
<?xml version="1.0" encoding="UTF-8"?>
<ESGMain xmlns="urn:dvb:ipdc:esg:2005" xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001"
xmlns:tva="urn:tva:metadata:2005" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >
   <ESG>
   <ContentTable>
      <Content contentID="cbms://foo.com/Content1">
         <Title xml:lang="en">Content Item 1</Title>
                  <CaptionLanguage closed='true'>en</CaptionLanguage>
         <CaptionLanguage closed='true'>fr</CaptionLanguage>
         <Duration>PT30M</Duration>
      </Content>
   </ContentTable>

   <ScheduleEventTable>
      <ScheduleEvent>
         <PublishedStartTime>2006-03-02T21:30:00</PublishedStartTime>
         <PublishedEndTime>2006-03-02T23:30:00</PublishedEndTime>
         <ServiceRef IDRef="cbms://foo.com/Channel1" />
         <ContentFragmentRef IDRef="cbms://foo.com/Content1" />
         <AcquisitionRef IDRef="cbms://foo.com/Acquisition/Event1" />
      </ScheduleEvent>
   </ScheduleEventTable>

   <ServiceTable>
      <Service serviceID="cbms://foo.com/Channel1">
         <ServiceName>Channel1</ServiceName>
      </Service>
   </ServiceTable>

   <AcquisitionTable>
      <Acquisition contentMimeType="video/H264" acquisitionID="cbms://foo.com/Acquisition/Event1" >
         <ComponentDescription>
            <ComponentCharacteristic xsi:type="VideoComponentType">
               <CodecCharacteristic>
                  <Codec href="urn:dvb:CS:Video:4.5.6"/>
               </CodecCharacteristic>
               <FrameRate>25</FrameRate>
            </ComponentCharacteristic>
            <ComponentCharacteristic xsi:type="AudioComponentType">
               <Codec href="urn:dvb:CS:Audio:4.2"/>
               <Language>en</Language>
            </ComponentCharacteristic>
            <SessionDescription xsi:type="SDPRefType" >
               <SDPStream>
                  <![CDATA[v=0
o=foo.com 751092616 751111042 IN IP6 30:200::13:3DF0
s=SDP Delivery for Channel1
t=0 0
a=flute-tsi:23498
a=flute-ch:1
a=source-filter: incl IN IP6 * 30:200::3DF0
m=application 12345 FLUTE/UDP 0
c=IN IP6 FF15::84:349D
]]>
                  </SDPStream>
               <SDPURI>urn:dvb:ipdc:esg:sdp</SDPURI>
            </SessionDescription>
         </ComponentDescription>
      </Acquisition>
   </AcquisitionTable>
</ESG>
</ESGMain>
```

Note that the ESG instantiation above gives several levels of indirection to the SDP file that are needed by the terminal to consume the multimedia session. Below is an example of such an SDP file when 3GPP Timed Text is used:

```
v=0
o=IPDC 2890844526 2890842807 IN IP4 126.16.64.4
s=IPDC SDP Subtitling example
i=An example of session description for a multimedia content with subtitles
c=IN IP6 FF1E:03AD::7F2E:172A:1E24
b=AS:77
t=3350323800 3350331000
a=source filter: incl IN IP6 * 2001:210:1:2:240:96FF:FE25:8EC9
a=min-buffer-time:500
m=video 4002 RTP/AVP 96
b=TIAS:62000
b=RR:0
b=RS:600
a=maxprate:17
a=avg-br:48000
a=rtpmap:96 H264/90000
a=framerate:25
a=fmtp:96 profile-level-id=42A01E; packetization-mode=1;sprop-parameter-sets=Z0IACpZTBYmI,aMljiA==
a=label:video_stream_1
m=audio 4004 RTP/AVP 98
b=TIAS:15120
b=RR:0
b=RS:600
a=maxprate:10
a=avg-br:14000
a=rtpmap:98 AMR/8000
a=fmtp:98 octet-align=1
a=lang:en
a=label:audio_stream_1
m=video 4006 RTP/AVP 102
b=TIAS:10000
b=RR:0
b=RS:1
a=maxprate:3
a=avg-br:5000
a=rtpmap:102 3gpp-tt/1000
a=fmtp:102 tx=20;ty=200;width=200;height=50;tx3g=aMljiA==,BY;max-w=720;max-h=576
a=lang:en
a=label:video_stream_2
m=video 4008 RTP/AVP 104
b=TIAS:10000
b=RR:0
b=RS:1
a=maxprate:3
a=avg-br:5000
a=rtpmap:104 3gpp-tt/1000
a=fmtp:104 tx=20;ty=200;width=200;height=50;tx3g=aMljiA==,BY;max-w=720;max-h=576
a=lang:fr
a=label:video_stream_3
```

The terminal may select the subtitling stream based on user preference or choice. The language indication can also be extracted from the media-header box of the 3GPP timed text data.

When acquiring the SDP, the terminal can determine which subtitling option is used from the following SDP information:

- In the case of 3GPP Timed Text format:
    - media type in the media line (m=) is set to "video";
    - media sub-type in the corresponding attribute line is set to "3gpp-tt".
- In the case of DVB bitmap format:
    - media type in the media line is set to "data";
    - the corresponding attribute line holds a "ts-content" attribute to signal the subtitling stream is transported in a Transport Stream which is encapsulated in RTP. The value of this parameter is set to "DVB-Subtitles".

5.2.2 Subtitle scaling

In this clause, an example of the text-based subtitling scaling is given. Figure 7 shows the configuration of the subtitling position at the service provider side.

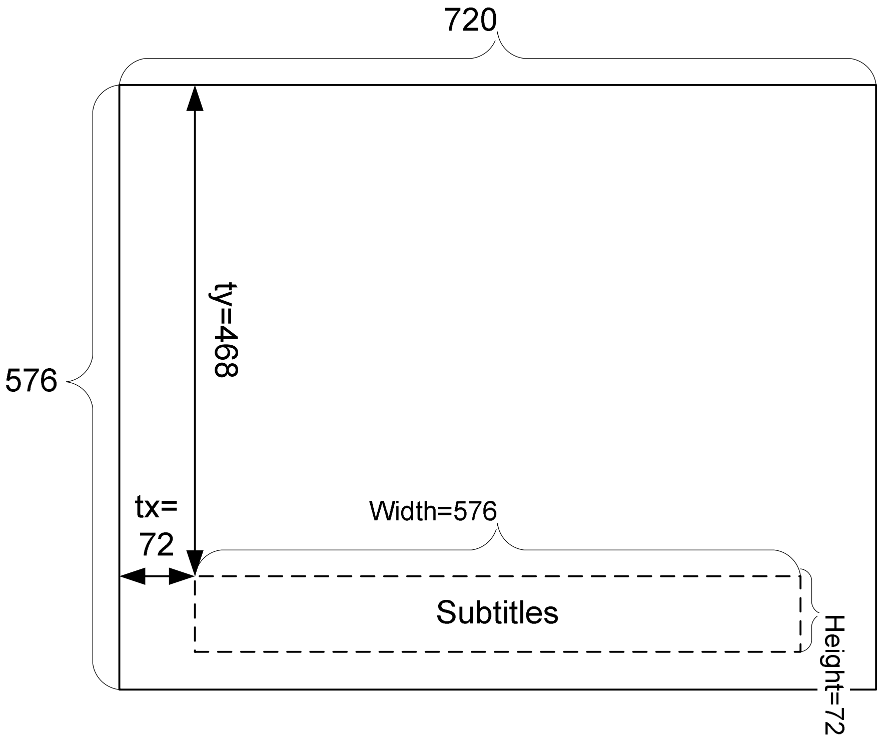

Figure 7: Position of the subtitles box within the video display area

The subtitling generator should maintain the same aspect ratio between the resolution of the video stream and the reference display area used for the scaling of the subtitling box. Assuming that the receiver the terminal has a display resolution of 320 x 240 pixel, the receiver will perform a scaling operation to calculate the position of the subtitling box within the display screen as well as the scaled font size. This is performed for the example given by figure 7 as follows:

1) Calculate the display area of the video stream after scaling according to the current terminal display. In this example we assume that a scaling is performed to maintain the aspect ratio of the original video. As a result, we assume that terminal displays the unused screen area with black bands on both sides. In our example, since the aspect ratio of the video is 5:4 whereas the display has an aspect ratio of 4:3, the actual video display area would then be 300 x 240 pixel. This operation is purely terminal specific and the terminal may decide to do another scaling that may or may not maintain the original aspect ratio.

2) Calculate the horizontal and vertical scaling ratios: Sx = 300 / 720 = 0,4166 and Sy = 240 / 576 = 0,4166.

3) Calculate the new subtitles box position and dimensions as follows: tx_scaled = tx × Sx = 72 × 0,4166 = 30, ty_scaled = ty × Sy = 468 × 0,4166 = 195, width_scaled = width × Sx = 576 × 0,4166 = 240, height_scaled = height × Sy = 72×0,4166 = 30.

4) Finally, based on the subtitles box dimensions (in this case 240×30) and the font type, the terminal selects the font size and adjusts the text so that it fits into the subtitles box.

5.3 Dynamic zapping service

A dynamic zapping service is a streaming service on a separate IP flow in a separate zapping burst, which contains complementary content about its associated main streaming service. The content within every zapping service burst may be dynamically updated.

Implementation guidelines on transmission of a dynamic zapping service over DVB-H networks can be found in TR 102 377 [i.4].

5.3.1 Delivery protocol

The following data types are available for the dynamic zapping service, all types are carried over RTP:

- Video: This is a video track related to an AV service. The quality of the video track may be reduced. This zapping type is transported over the baseline IPDC protocol stack as specified for the real-time streaming video service [2].

- Audio: This is an audio track related to an AV service. The quality of the sound may be reduced. This zapping type is transported over the baseline IPDC protocol stack as specified for the real-time streaming audio service [2].

- DynamicText: The dynamic text may contain content related to the progress of the associated AV service, such as sub-titling, or other textual content. This zapping type is transported over the baseline IPDC protocol stack as specified for Subtitling using 3GPP Timed Text Format [2].

5.3.2 Session description with SDP

The streaming session description with SDP for the zapping types Video, Audio and DynamicText is used as specified in [2].

Below is an example of an SDP file when one video with 2 pictures per second, one monaural audio track and DynamicText is used. The buffering time is set to 0 for minimum delay until presentation.

```
v=0
o=IPDC 2890844526 2890842807 IN IP4 126.16.64.4
s=IPDC SDP example dynamic zapping service
i=An example of session description for a dynamic zapping service with video, audio and dynamic text
c=IN IP6 FF1E:03AD::7F2E:172A:1E24
b=AS:77
t=3350323800 3350331000
a=source filter: incl IN IP6 * 2001:210:1:2:240:96FF:FE25:8EC9
a=min-buffer-time:50
m=video 4002 RTP/AVP 96
b=TIAS:62000
b=RR:0
b=RS:600
a=maxprate:8
a=avg-br:32000
a=rtpmap:96 H264/90000
a=framerate:2
a=fmtp:96 profile level id=42A01E; packetization mode=1;sprop parameter sets=Z0IACpZTBYmI,aMljiA==
a=label:video_stream_1
m=audio 4004 RTP/AVP 98
b=TIAS:5720
b=RR:0
b=RS:600
a=maxprate:4
a=avg-br:5200
a=rtpmap:98 AMR/8000
a=fmtp:98 octet align=1
a=lang:en
a=label:audio_stream_1
m=video 4006 RTP/AVP 102
b=TIAS:10000
b=RR:0
b=RS:1
a=maxprate:3
a=avg-br:5000
a=rtpmap:102 3gpp tt/1000
a=fmtp:102 tx=20;ty=200;width=200;height=50;tx3g=aMljiA==,BY;max w=720;max h=576
a=lang:en
a=label:video_stream_2
```

5.4 Hybrid Streaming Delivery

5.4.1 Introduction.

A hybrid Mobile TV system is a mobile media delivery system where services are available over both a broadcast bearer (such as DVB-H) and a unicast bearer (such as 3G) at the same time. Hybrid delivery is of interest for one or more of the following reasons:

- Due to coverage, broadcast access is not available in some geographical areas (or indoor reception).
- Due to mobility, broadcast access is not available in some service areas.
- Unicast access may be cheaper than to introduce gap fillers to reach coverage.
- Only the most popular services can be made available over broadcast delivery. Those services which cannot fit into the broadcast delivery can be made available over unicast.

Switching between different bearer types is sometimes referred to as near seamless bearer handover. The tern "near" is included as due to the nature of the switching there may occur some blackout period until the service can be rendered using data transmitted over the new bearer.

5.4.2 Reference Scenario

The hybrid scenario outlined here uses a PSS streaming session over unicast access in addition to the broadcast access.

The specification of Packet Switched Streaming (PSS) is defined in [i.6] and allows the following media types to be controlled using RTSP methods:

- Video
- Audio
- Timed Text

Thus it is possible to move these media from a DVB-H broadcast access to a PSS session and vice-versa.

In this reference scenario the PSS session is setup to the service layer at the start of a Mobile TV session. Then, if the client can access the requested service over the broadcast bearer the PSS session is left in a paused state. The only signaling between the PSS client and the PSS server is "keep alive" messages to prevent the server to time-out.

If broadcast coverage is lost the Mobile TV application will continue the service reception using the unicast access. This is done by issuing an RTSP PLAY to the streaming server. When broadcast coverage later is reacquired an RTSP PAUSE should be issued to stop unicast reception and the broadcast access will again be used.

Using the IPDC ESG it is possible to have Service Fragments or Schedule Event Fragments referencing multiple Acquisition Fragments. One Acquisition Fragment may contain a session description for the broadcast access and another Acquisition Fragment may contain a session description of "PSSAccessType" for the unicast access. It should be understood that the two Acquisition fragments actually refer to the same program.

5.4.3 Access Switching

In this clause some possibilities for the handover of a streaming session between different access technologies are discussed.

Two types of access change can occur: unicast to broadcast and broadcast to unicast.

5.4.3.1 Broadcast to Unicast Switch

Typically this access change occurs suddenly and without warning when the available signal strength is no longer sufficient for broadcast access. The client may identify the loss of broadcast access by signal strength measurements, that packet losses exceed a certain threshold, or that RTP reception has stopped completely. As a result the client may terminate the broadcast session and initiate reception over the interaction channel instead. It might be up to the user to decide whether the switch of bearer is desired or if the session shall be terminated, especially if delivery over the interaction channel induces additional monetary cost.

During the change of access to unicast delivery it may be that a drop-out of the service will last corresponding to the time it takes the client to setup reception over unicast access, a possible change of codecs, and to fill its receiver buffers to a suitable level. This could be in the order of several seconds.

As was previously mentioned, this scenario uses a unicast PSS session that is set up in parallel to the broadcast session at the initialization of the mobile TV client application. In RTSP terms, the required number of SETUP methods has already been exchanged between the client and the server and the PSS session is left in a PAUSED state. Thus, when broadcast coverage is lost, the client simply requests RTSP PLAY on the server for unicast access. Using such a PAUSED PSS session is one method to shorten the black-out period compared to if a new session needs to be negotiated at the time of loss of broadcast signal

Of course, it may happen that the user has changed service since the initial setup such that the unicast session that was setup no longer refer to the broadcast session being received. Using the "Fast Channel Switching" framework defined in [PSS] the client may issue a PLAY with the URL for a unicast session that corresponds to the lost broadcast session.

Additionally it may be that the PSS service layer provides "network based time-shifting" as defined in [i.6]. When both the service layer and the client support Range headers expressed in UTC it is possible to resume the session over unicast at the time instant where the BC access was lost. Thus there may be a blackout during the access change but there would be no loss of media.

Utilization of the different access types may imply that an access change also results in usage of a different set of codecs or different codec setting. In such a case a client needs to prepare a new set of decoders and possibly having them run in parallel for a short period of time. A reason for changing the codec or its settings could be due to a larger change of the available bitrate.

5.4.3.2 Unicast to Broadcast Switch

If the client is receiving the service over the unicast channel, and it recognizes that the broadcast channel is re-gained, it may terminate or pause the unicast PSS session and initiate reception over the broadcast channel.

An access change from UC to BC has the possibility to be made seamless when both BC and UC streams can be received in the client at the same time.

As for the broadcast to unicast switch, new codecs or new codec settings may be needed.

5.4.3.3 Usage of SSRC

One of the challenges of making an access change of a service, being either a change from broadcast to unicast or a change from unicast to broadcast, is to align the media flows at the point of change while keeping a good user experience. There are some possibilities to make this alignment at the RTP level:

1) For each media the client compares the SSRC values of the unicast and broadcast flows and detects two different SSRC values. The client assumes that the same wall clock time is used for the two flows but needs to waits for flow specific RTCP packets to be received in order to align the streams using the wall clock time.

2) For each media the client compares the SSRC values of the unicast and broadcast flows and detects that the same SSRC value is used. The client assumes that the same random RTP timestamp offset is used for the flows and align the streams using the RTP timestamp.

Of course, the most straight forward way for the client to align two flows would be if they are encoded using the same codec and codec settings. In addition, if the SSRC of the two flows are the same, as described in 2) above, it would further help the client to make a flow change with good user experience.

6 Data download to devices

This clause gives guidelines on the use of file delivery enablers as specified in [2] for data download services. The setup and operation of a file delivery session is described in clause 6.1. Examples of the use of post-delivery procedures are covered in clause 6.3. In clause 6.4 the use of application layer FEC is discussed with recommendations on its use. The specific aspects to consider in Webcasting service are covered in clause 6.5. Finally, clause 6.6 highlights the terminal capability issues to consider when operating file delivery services.

6.1 Setup and operation of a file delivery session

This clause describes the steps to setup and operate a file delivery session.

6.1.1 Session initiation using the ESG

This clause describes the parameters that are extracted from the ESG and passed to the application to initiate a file delivery service.

6.1.1.1 File delivery session discovery

A file delivery service is described in the ESG by a Service Fragment. The contents of the file delivery service are described each by a Content Fragment. The user selects the services he wants to consume and may perform a purchase step. Thereafter, the terminal locates the corresponding acquisition fragment by looking for the referring Schedule Event fragments and/or Service Fragment. The user may be offered to select the reception time, if more than one Schedule Event fragment for the same content exist.

Upon locating the corresponding Acquisition Fragment, the terminal checks the Component Characteristics for the file format and the storage requirements. It acquires also the SDP of the FLUTE session, whether it is embedded or delivered out-of-band. The SDP of the FLUTE session indicates the source IP address, the TSI, and the destination IP address and port number for each of the channels. This information is given to the FLUTE receiver to join the FLUTE session.

6.1.1.2 FLUTE session initiation

In order to initiate a FLUTE session the terminal must know:

- The parameters of the session.
- Optionally, the files to be retrieved.
- The timeframe of the session.
- The IPPlatformID in use, in order to perform IP address resolution at PSI/SI level. This information is usually available from the bootstrap phase of the DVB-H device, where it selects the IP Platform over which it will operate.

A FLUTE session is made of one or more IP Multicast streams transporting LCT data. Therefore, the terminal must tune to one or more of these Multicast streams in order to participate in the FLUTE session.

The parameters of the session are provided in the Session Description of the Acquisition fragment in the ESG. See clause 5.1.1.1.1 for detailed description of the alternatives for SDP retrieval and delivery.

The session description include the following information:

- Network layer information:
  - Source IP address from the source-filter attribute.
  - Destination IP address from the "c=" field. Note that if many LCT channels are available ("flute-ch" attribute), destination IP address for each LCT channel can be declared either on session level or media level. It is recommended that the "c" field is included in each media description indicating the destination IP address of each LCT channel.

- Transport layer information:
  - Destination port from the port subfield in the "m=" line. Note that if many LCT channels are available ("flute-ch" attribute), for each LCT channel a media line is declared in SDP which also indicates the destination port.
- Application transport layer (ALC/LCT):
  - The TSI of the session from the "flute-tsi" attribute.
  - The FEC capabilities, depending on the FEC scheme in use.

A FLUTE session may consist of multiple LCT channels. Usage of multiple channels is optional for network and terminals.

The terminal detects that a FLUTE session uses multiple channels from the Acquisition fragment of the ESG, by checking the number of ComponentCharacteristic elements of type FileDownloadComponentType that are associated to the SessionDescription that refers to the given FLUTE session , and also from the "a=flute-ch" attribute in the SDP file of the FLUTE session.

Additionally, the FileDownloadComponent carries an indication of the bandwidth requirement of the FLUTE channel, its purpose, the file formats of the files and the storage requirement for receiving the files of the channel. Note, however, that there is no explicit link between a ComponentCharacteristic element and a LCT channel of the FLUTE session.

Therefore terminals are not provided with enough information which will enable them to decide which specific channel (s) of the FLUTE session they should join to fetch a specific content. Hence, the FLUTE sender should assume that terminals will join the base channel, which is the first channel that appears in the SDP file (first "m=" line). The FLUTE sender should make sure that terminals are able to correctly receive all the content that is announced in the ESG by tuning to the base channel only.

6.1.1.3 Delivered file identification

When joining the FLUTE session, the application indicates to the FLUTE receiver the items that the user wishes to download from that session. Each file is identified by the URI which corresponds both to the Content Location in the Schedule Event fragment and the Content-Location element in the FDT of the FLUTE session.

Even if only a single file is carouselled during the timeframe indicated in the Schedule Event, the inclusion of the ContentLocation element is needed to simplify processing in the terminal and undesired acquisition of objects scheduled immediately before or after the indicated time, especially when clock synchronization is not perfect.

The FLUTE receiver identifies the corresponding TOI value for each of the requested files. The FLUTE receiver identifies the most up to date TOI value by retrieving the FDT with the highest FDT Instance ID that lists the requested file.

The CDP specification also defines a method to indicate groups of files. The FLUTE receiver will retrieve all files of a group, if instructed to retrieve at least one file that belongs to that group.

6.1.1.4 Determining MIME type of received files

When discovering a file delivery session from the ESG, it is recommended that:

- the ContentLocation element in ScheduleEvent Fragment signals the entry point for the application;
- the contentMimeType attribute within the AcquisitionType is used to determine the primary MIME type of the file serving as entry point to be delivered within a given session and to determine the application that is able to consume the files.

It is recommended to use the MIME types registered by IANA [19].

As an example in a Webcast service the ContentLocation element may be set to "http:/foo.htm" and the contentMimeType may be set to "text/html" to indicate that the entry point is a HTML page, even though a large number of the files delivered may be images used within the HTML page.

NOTE: This recommendation assumes that all entry points in the references to the Acquisition Fragment are of the described MIME type.

Ultimately only the FDT provides the MIME type associated with each file delivered through the Content-Type attribute, which is mandatory for FLUTE when used in IP Datacast over DVB-H.

6.1.1.5 Timing parameters

The timing parameters include the start and end time of the session. This information is provided in the ScheduleEvent fragment with the PublishedStartTime and PublishedEndTime elements and in the "t=" line in the SDP.

The user should only see the timing published in the ESG and select content based on this timing. The FLUTE receiver is expected to consume the sdp delivered along with the ESG. The FLUTE sender should not expect that the terminal will override the sdp timing information based on the ESG information, therefore the sdp should have some relaxed session timing boundaries (i.e. larger than the timing boundaries in the ESG). However, the FLUTE sender should not expect that terminals start reception before the start time published in the ScheduleEvent fragment.

6.1.2 FLUTE operation

This clause illustrates the FLUTE protocol operation.

Figures 8 and 9 describe the operation of an IPDC receiver. The terminal performs the steps described in the SDL diagram for each received Transport Object. An FDT may include a number of transmission objects, each identified by a unique Transmission Object Identifier (TOI).

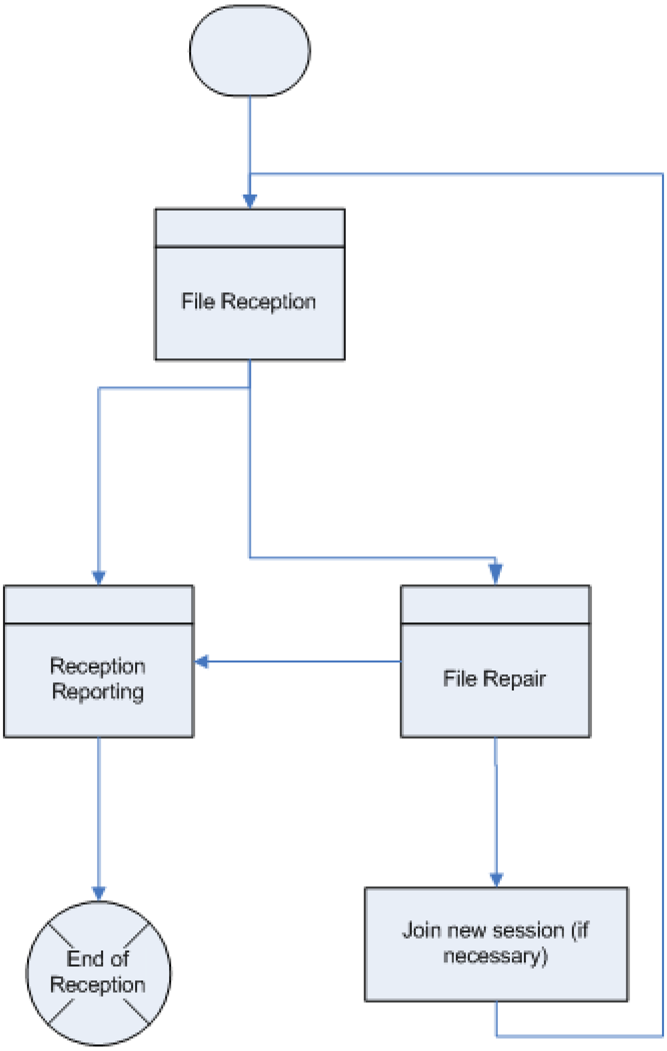

Figure 8: File Reception in IPDC - Overview

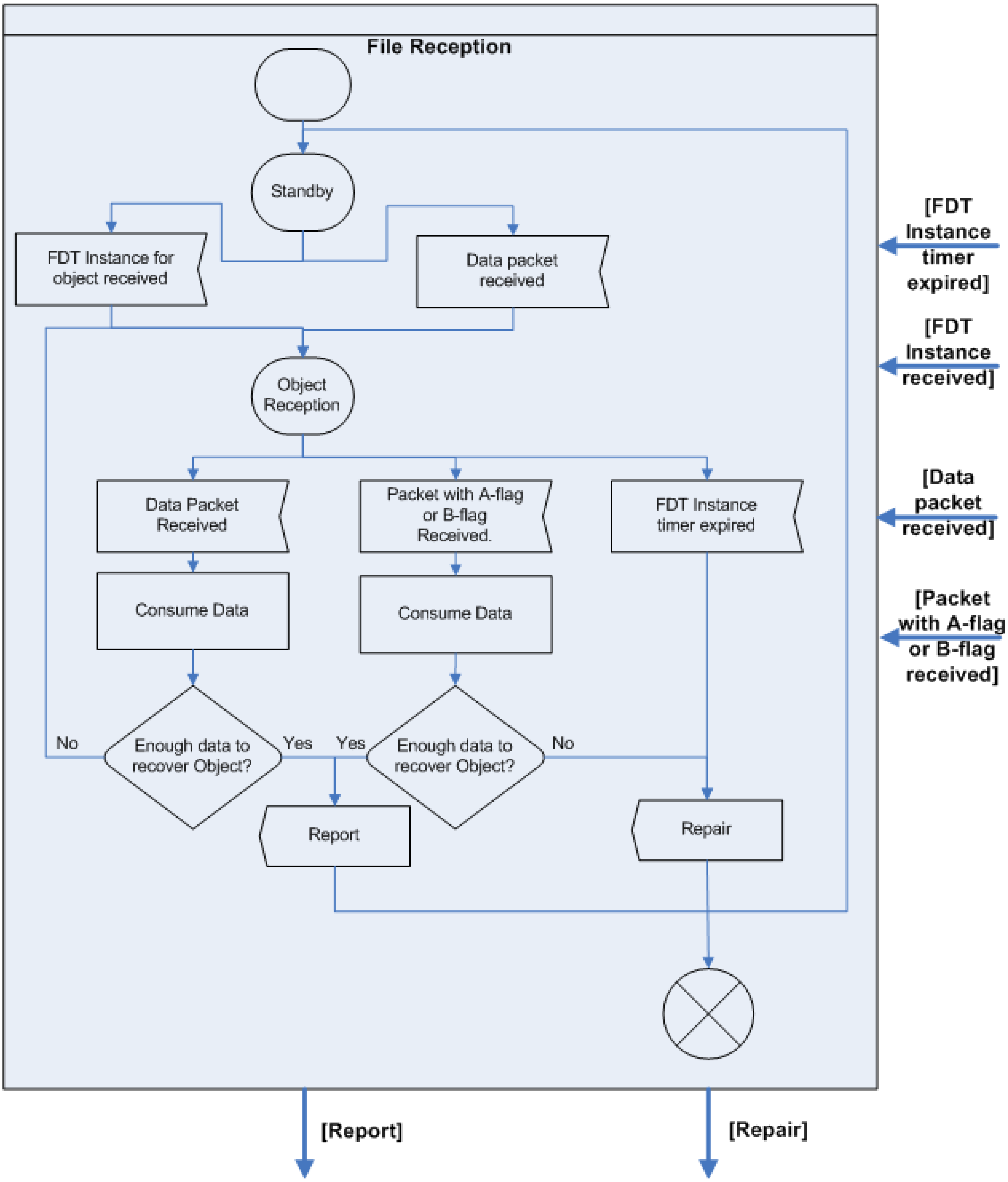
File Reception
Standby
FDT Instance for object received
Data packet received
Object Reception
Data Packet Received
Packet with A-flag or B-flag Received.
FDT Instance timer expired
Consume Data
Consume Data
Enough data to recover Object?
Enough data to recover Object?
No
Yes
Yes
No
Report
Repair
[FDT Instance timer expired]
[FDT Instance received]
[Data packet received]
[Packet with A-flag or B-flag received]
[Report]
[Repair]

Figure 9: File Reception

6.1.3 Handling time issues in FLUTE

During the operation of a file delivery using FLUTE, the terminal needs to keep accurate time synchronization with the FLUTE sender. This is necessary in order to correctly perform several operations such as the detection of the FDT expiry time, which is delivered as an NTP timestamp according to the sender time. The detection of the session end as well as the initiation of the associated delivery procedures also rely on synchronized terminal clock to function properly.

The terminal should use the Time and Date Table (TDT) as specified in [9] to extract the current network time information. The TDT is delivered in the PSI/SI at least once every 30 s as specified in [5]. The TDT provides the current time in MJD (modified Julian date) and UTC (universal time, co-ordinated).

The terminal needs to perform a mapping between NTP timestamps, which are expressed in the number of seconds since 0000 01.01.1900 and the UTC time extracted from the TDT. How this is performed is outside of scope of the present document.

The FLUTE sender should synchronize its sender clock to the clock used by the DVB network. For this purpose an NTP or SNTP service may be used.

6.2 Post-Delivery procedures

6.2.1 File repair

File repair mechanism is defined in the context of the Associated Delivery Procedures, related to a file delivery session. So, files that may need to be repaired are typically some files previously broadcasted in the related session using the transport protocol FLUTE. This protocol (based upon the ALC protocol) enables transport of files using a set of related packets.

6.2.1.1 File repair example

The following steps are describing such an example of the File Repair procedure (based upon Point to Point connection only during repairing), in a chronological order. It begins with the announcement of File Repair capability, related to the current Delivery Session, through an adequate XML file. Then, assuming a receiver has not received a complete file from this session, a file repair procedure is engaged, using a file repair service. Some messages that are exchanged between the receiver and the repair service (to achieve repairing of the file) are shown. The repair procedure is achieved once the receiver obtains from repair service all the missing parts of the file.

6.2.1.1.1 Announcement of file repair capability

File repair capability related to a delivery session has to be previously signalled to receivers, to enable these receivers to possibly initiate such a file repair activity. As described in clause 7.2 of TS 102 472 [2], the format for an instance of configuration of an associated delivery procedure is an XML file. This file may be procured in-band within the delivery session, or out-of-band (within the ESG, before start of delivery session).

Inside that XML file configuration of the associated delivery procedure is done using an "associatedProcedureDescription" element, which in turn contains a "postFileRepair" element to describe such a file repair capability. This element is listing the repair service(s) that receivers may contact for file repair activity, using one "serverURI" element for each useable repair service. Such an element indicates the URI (as a name or as an IP address to prevent DNS process) of the related repair service.

6.2.1.1.2 Example of an associated delivery procedure configuration file

Here, three repair service URIs are listed, inside the postFileRepair information:

```
<?xml version="1.0" encoding="UTF-8"?>
<associatedProcedureDescription xmlns="urn:dvb:ipdc:cdp:associatedProcedures:2005"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:dvb:ipdc:cdp:associatedProcedures:2005
associated-procedure-description.xsd">
<postFileRepair
offsetTime="50"
randomTimePeriod="600">
<serverURI>http://ipdcrepair.operator.umts/ipdc_file_repair_script</serverURI>
<serverURI>http://ipdcrepair1.operator.umts/ipdc_file_repair_script</serverURI>
<serverURI>http://ipdcrepair2.operator.umts/ipdc_file_repair_script</serverURI>
</postFileRepair>
<bmFileRepair sessionDescriptionURI="http://www.example.com/ipdc/session1.sdp"/>
</associatedProcedureDescription>
```

6.2.1.1.3 Recommendations for the file delivery server and the repair server

The current CDP specification allows much freedom to the repair server on the order in which it sends the repair symbols and on the presence or absence of padding bytes in the response. In order to allow easy parsing of the repair response by the client, it is strongly recommended that:

- The file delivery server includes the Transfer-Length attribute in the FDT, if ever content encoding is used (i.e. if the Content_Length differs from the Transfer-Length).

- If the last symbol has a length different from the preceding source symbols the repair server pads the last symbol with zero-bytes to the same length as other symbols.

- The repair server may signal the end of the response (payload) with two bytes of 00.

Note that the repair server is not required to deliver the requested symbols in the same order as the request.

6.2.1.1.4 Context of file delivery

We assume for this example that a file with URI "www.news.ipdc.com/latest/ipdcFileTest.txt" is broadcasted, during the current file delivery session, for which the file repair procedure has been defined (see above). That file is assumed to be transmitted using FLUTE with following characteristics and parameters:

This file is 199 497 bytes long, and:

- Compact No-Code FEC used,
  Each FLUTE encoding symbol length: 500 bytes,
  Maximum allowed source block length = 100 encoding symbols,
  3 sources Blocks (50 000 bytes long) each containing 100 symbols, and the fourth containing only 99 symbols. The last symbol of this last source block will be padded by the repair server to get full length (i.e. same length as the other symbols).

- A receiver has previously received a related FDT, announcing next broadcasting of the file *ipdcFileTest.txt* with a particular TOI.

End of file broadcast is detected by the receiver either through reception of last object packet with close object flag, or if this object is lost through the timeout of the wait timer.

6.2.1.1.5 Detection of file repair need

The receiver tries to build the original file and finds it to be incomplete. Some source symbols are missing:

SBN: 0; ESI=12,44,78

SBN: 2 (all the 100 symbols of source block 2 are missing)

SBN: 3; ESI=55-98 (here: a range of missing symbols)

6.2.1.1.6 Initiation of file repair procedure

After a wait (based upon "offsetTime and randomTimePeriod" parameters from XML configuration file previously received), the receiver tries to initiate a repair procedure, using a randomly chosen repair server from the list that was specified (again in XML configuration file).

The randomly chosen repair service is "ipdcrepair2.operator.com" in our example. The relative path of the repair script to be used on this service for repair activity is assumed to be: "/ipdc_file_repair_script".

The Receiver connects to this service using a TCP connection.

6.2.1.1.7 Exchanges between receiver and repair service

The receiver submits the repair request using the HTTP protocol:

- GET /ipdc_file_repair_script?fileURI=www.news.ipdc.com/latest/ipdcFileTest.txt&SBN=0;ESI=12,44,78&SBN=2&SBN=3;ESI=55-98 HTTP/1.1.

- Host: http://ipdcrepair2.operator.com.

We assume that the repair server is using only point to point repair strategy, and that it has access to all the missing symbols that are claimed by the receiver. The repair server sends its answers in the same TCP/HTTP session as the request, starting with an HTTP header (see below), followed by packets of claimed symbols, but not necessarily in the order of the request.

The HTTP response message structure is:

HTTP Header	**HTTP/1.1 200 OK Content-Type: application/simpleSymbolContainer**		
Structure of a group of responses	**Number of symbols in this group (2 bytes, in network order)**	**Fec Payload ID (4 bytes) (SBN,ESI)**	**Encoding symbol(s)**
1st group	1	(0 , 12)	[XXXX]
2nd group	1	(0 , 44)	[YYYY]
3rd group	34	(3 , 65)	[cccc] [dddd], etc.
4th group	100	(2 , 0)	[AAAA] [BBBB] [CCCC] [DDDD], etc.
5th group	1	(0 , 78)	[ZZZZ]
6th group	10	(3 , 55)	[aaaa] [bbbb], etc.
last group	00	<inclusion of this last group is optional>	
NOTE: [XXXX], [YYYY], etc.: this syntax represents data related to an encoded symbol sent in payload of HTTP service's response.			

For repair response message content encoding should not be applied as its use is ambiguous.

6.2.1.1.8 Example of receiver behaviour

The receiver will calculate from the information available in the FDT the size of the repair symbols, the total number of symbols in the file, the FEC Payload ID (SBN, ESI) of the last symbol, and the size of this last symbol.

After receiving the http response header it will parse the payload in the following loop:

- Read the two bytes of "number of symbols".

- Break the loop if this number is 0.

- Read the 4 bytes of "FEC payload ID".

- Calculate the length of the group by multiplying the number of symbols by the symbol length.

- Read the rest of the group (encoding symbols).

The receiver should remove the padding bytes before further processing. The described procedure allows protocol extensions and permits reception of partial repairs without relying on time-outs or connection loss.

6.2.1.1.9 Use of chunked transfer encoding

The HTTP1.1 specification [10] allows the pipelining of multiple requests to the same server on a single TCP connection. In order to allow for reliably identifying the end of a message body, chunked transfer encoding should be applied in this case. The following example represents the same HTTP response as above, using chunked encoding.

NOTE: Normal type face indicates ASCII characters, the sign ↵ indicates the sequence of carriage return - line feed characters, hexadecimal notation within brackets in italic font is used to indicate octet streams.

```
HTTP/1.1 200 OK↵
Content-Type: application/simpleSymbolContainer↵
Transfer-Encoding: chunked ↵
↵
1FA↵
[00][01][00][00][00][0C][..][..] ……………… [..]↵     <6+1*500= 506 bytes + CR,LF>
1fa↵
[00][01][00][00][00][2C][..][..] ……………… [..]↵     <6+1*500= 506 bytes + CR,LF>
426e↵
[00][22][00][03][00][2B][..][..]………………[..][00][00][00]↵
                          <6+33*500+497+3(padding bytes) = 17,006 bytes + CR,LF>
C356↵
[00][64][00][02][00][00][..][..] ……………… [..]↵     <6+100*500= 50 006 bytes + CR,LF>
1FA↵
[00][01][00][00][00][4E][..][..][..][..][..]↵     <6+1*500= 506 bytes + CR,LF>
138e↵
[00][0A][00][03][00][21][..][..]………………[..]↵       <6+10*500= 5 006 bytes + CR,LF>
2↵
[00][00]↵
0↵
↵
```

At this place the response to the next http request may be sent.

6.2.1.1.10 End of file repair procedure

Once the repair service has answered all pending requests and sent all the claimed encoded symbols, it may end the TCP connection with receiver. In order to allow the client to process the entire response and possibly to chain another request, the server should do so only after an appropriate timeout.

The client has received the missing symbols, and can achieve the related file completion. It can close the TCP connection immediately when it has no further requests.

6.2.1.2 Special considerations in case of file versioning

As defined in clauses 7.3.2 and 6.1.9 in [2], file repair is triggered by the end of file delivery or the end of session. End of file delivery is detected at the latest when the most recent FDT Instance that declares the file has expired or when the whole file delivery session has ended. Alternatively, the server can send a FLUTE packet with the A-flag or B-flag set to 1, which respectively indicate pre-mature end of session or end of file delivery.

If a file repair service exists, terminals that support file repair and identified the need to perform a file repair request will determine a time instant when they will send their repair requests. As described in clause 7.3.4 in TS 102 471 [3], this time is calculated based on an offset time and a random time which falls within a random time period. By consequence, the repair requests will be limited within a certain time interval. When more than one version of the file is sent in the file delivery session, the terminal does not have any means to indicate to the repair server, which version of the file the repair request relates to. On the other hand, the repair server cannot uniquely identify the version of the file to which repair requests relate, if the repair request periods for the different versions of the file do overlap (this scenario is depicted in figure 10).

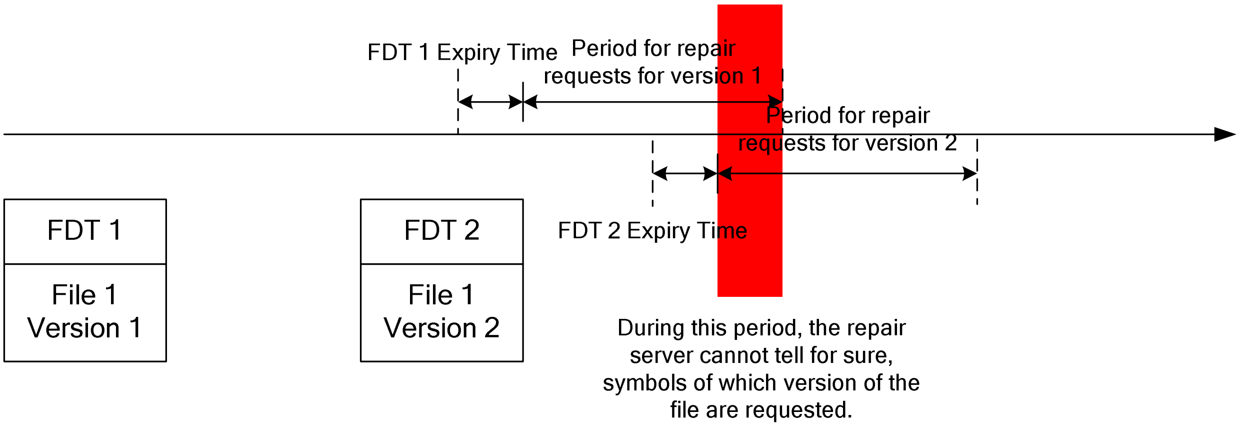
FDT 1 Expiry Time
Period for repair requests for version 1
Period for repair requests for version 2
FDT 2 Expiry Time
FDT 1
File 1 Version 1
FDT 2
File 1 Version 2
During this period, the repair server cannot tell for sure, symbols of which version of the file are requested.

Figure 10: Ambiguous file repair requests

On the service provider side, a way to avoid this situation is to set the triggers for the file repair procedures so that repair request periods for different versions of a file do not overlap. This can e.g. be achieved by setting the FDT expiry time of the FDT instance that declares version n of a file as follows:

$$\mathrm{FDT}_{\mathrm{expiry}}(n) = \mathrm{FDT}_{\mathrm{expiry}}(n-1) + \mathit{offsetTime} + \mathit{randomTimePeriod} + \Delta \qquad (5)$$

Where Δ is a protection time period that accounts for unsynchronized clock drifts and network delay jitters.

Such calculation can also be used when the server needs to make sure the use of the A or B flag will not cause file repair periods to overlap:

$$\mathrm{t}_{\mathrm{A\ or\ B\ set\ to\ 1}} = \mathrm{t}_{\mathrm{previous\ \ FDT\ \ expiry}} + \mathit{offsetTime} + \mathit{randomTimePeriod} + \Delta \qquad (6)$$

Figure 11 depicts the case where no overlap in the repair periods exists, so that the repair server can uniquely identify the version of the file for which repair requests are received.

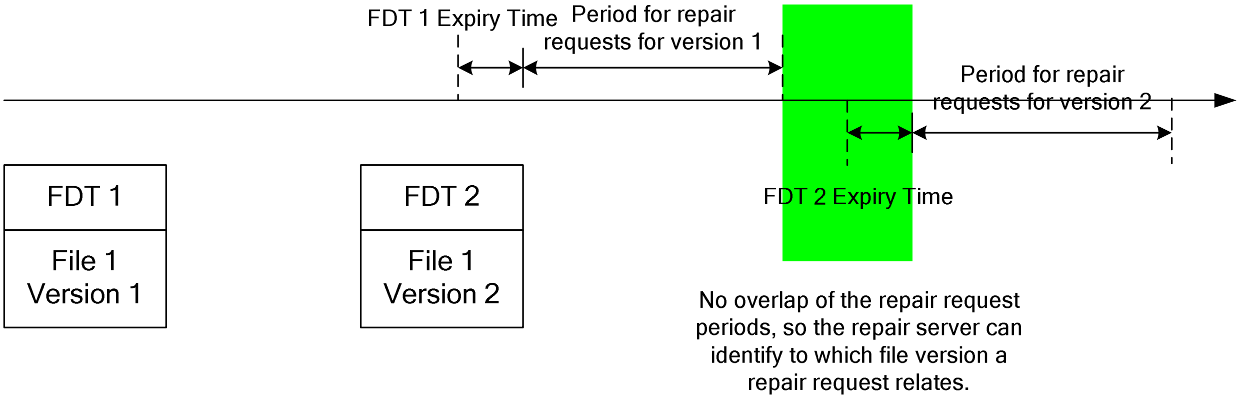
FDT 1 Expiry Time
Period for repair requests for version 1
Period for repair requests for version 2
FDT 2 Expiry Time
FDT 1
File 1 Version 1
FDT 2
File 1 Version 2
No overlap of the repair request periods, so the repair server can identify to which file version a repair request relates.

Figure 11: Unambiguous file repair requests

Additionally, the receiver can use the MD5 file digest to check the correctness of the file reconstruction after file repair, whenever the MD5 file digest is provided by the server in the FDT.

So if the FDT instance FDT1 was:

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<FDT-Instance xmlns="http://www.example.com/flute" Expires="3396186000">
<File Content-Location="cvs-tortoise.pdf" Content-Type="application/pdf" TOI="9" />
<File Content-Location="BMW.3gp" Content-Type="application/octet-stream" TOI="10" />
<File Content-Location="neoclip1.wmv" Content-Type="video/x-ms-wmv" TOI="11" />
<File Content-Location="mode d emploi Billet Electronique.pdf" Content-Type="application/pdf"
TOI="12" />
</FDT-Instance>
```

i.e. this FDT would expire the 15 August 2007, at 16 h 00:00 UTC.

If the file with the TOI 10 is to be superseded by a new version, this can be done (taken the values from the previous clause) at earliest 50 s (offstet time) + 10 min. (random time period) after the FDT expiry. The protection time period (delta) in this example is 10 s. So the updated file may be sent with the FDT:

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<FDT-Instance xmlns="http://www.example.com/flute" Expires="3396186660">
<File Content-Location="BMW.3gp" Content-Type="application/octet-stream" TOI="13" />
</FDT-Instance>
```

i.e. an expiry time of 15 August 2007, at 16 h 11 min UTC.

6.2.1.3 Hybrid file repair example

6.2.1.3.1 Use cases of file repair

The following is describing three possible use cases of file repair in CDP. In addition to point-to-point based file repair example described in clause 6.2.1.1, hybrid mode file repair is also possible. For better transmission efficiency or for some other reasons, file repairing can be done in mixing of P2P and P2M mode. Then, we can think about three use cases as follows;

- P2P only: If the repair server is requested only a few number of symbols by receivers, then sending all the repair symbols in HTTP response payload over unicast may be efficient. Moreover, the receivers that are in interactive network coverage while not in DVB coverage at the same time have no other choices except P2P access in file repairing.

- P2M only: If lots of terminals request common repair symbols or there already exist available P2M sessions having requested symbols, only using P2M may be enough in file repairing.

- Hybrid: Mixing of P2P and P2M way is also possible in repairing a file. In this case, some parts of the requested encoding symbols/source blocks would be delivered in P2P way, while the others are expected to be delivered by redirecting to a P2M session.

6.2.1.3.2 Repair server behaviour

In repair server side, the server can be in one of three basic server modes; P2P only, Redirection only and Hybrid mode. The repair mode of each server can be specified in AssociatedDeliveryProcedureDescription file as service URL attribute. After receiving repair request messages from receivers, the server operates in its own mode. Figure 12 shows the example behavior of repair server.

In P2P only mode, the server always provides all the symbols requested by the terminals over interactive channel. This mode is needed for the terminals that do not have DVB connection and it should be signaled that such server has P2P only capability.

The server may just only redirect to another server or to the P2M bearer session. In this case, associated repair service URL used by the terminal may not have any capability signaled as its attribute.

In hybrid mode, based on the terminal requests and possibly some system state, the server can decide whether to deliver all the missing symbols in P2P or redirect either to P2M (as in annex B of [2]) or another P2P server. In this case, associated repair service URL used by the terminal would not have any capability signaled as its attribute, too.

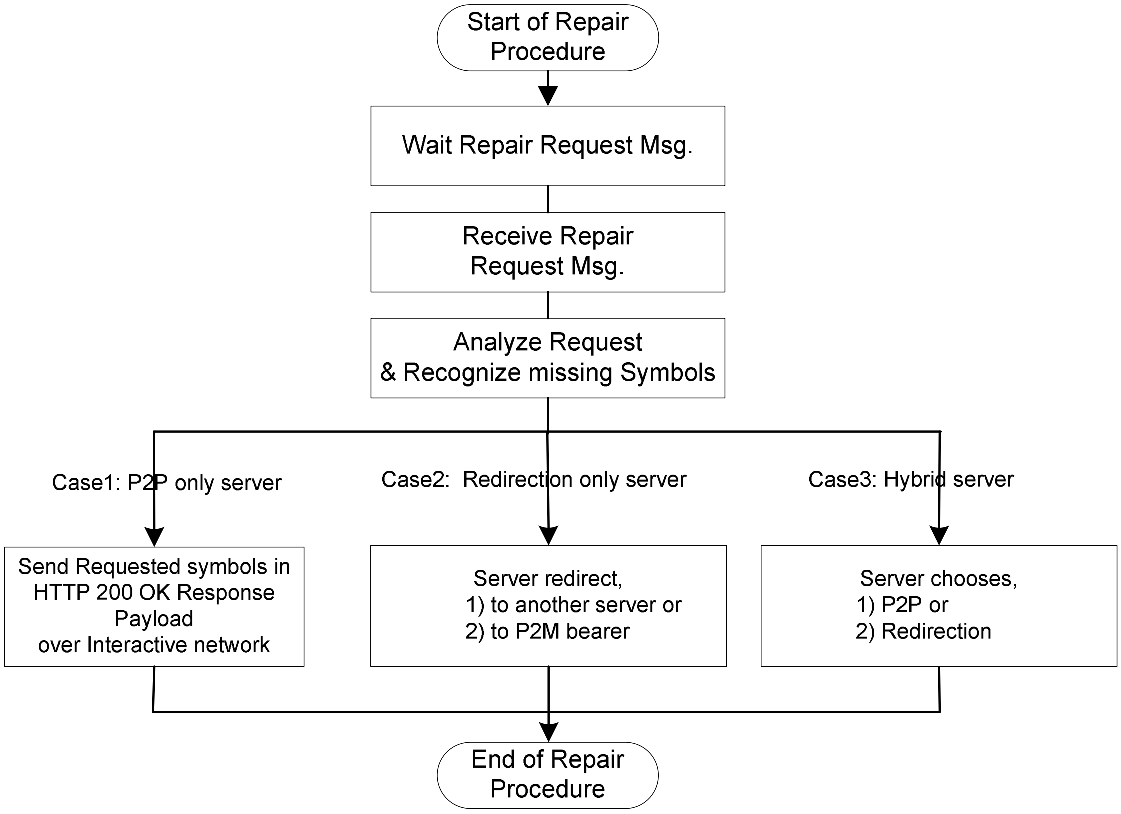
Start of Repair Procedure
Wait Repair Request Msg.
Receive Repair Request Msg.
Analyze Request & Recognize missing Symbols
Case1: P2P only server
Case2: Redirection only server
Case3: Hybrid server
Send Requested symbols in HTTP 200 OK Response Payload over Interactive network
Server redirect, 1) to another server or 2) to P2M bearer
Server chooses, 1) P2P or 2) Redirection
End of Repair Procedure

Figure 12: Example of Repair Server Behaviour

6.2.1.3.3 Receiver behaviour

After the receiver detects the need of file repair, it will send HTTP file repair request message to the server. Before sending request message to server, the receiver selects file repair server in the consideration of its repair capability attribute. When no server capability is specified, terminal should assume that server may redirect it to P2M repair. Figure 13 shows the example behavior of receiver.

For the receivers that is in coverage of interactive network while not in that of DVB, it should select P2P only mode repair server first. It is because that, some server may redirect requests to P2M session bearer. Therefore, it is recommended that at least one P2P repair server is explicitly signaled in AssociatedDeliveryProcedureDescription.

It is up to the terminal to reiterate the requests to repair servers until all missing symbols have been received.

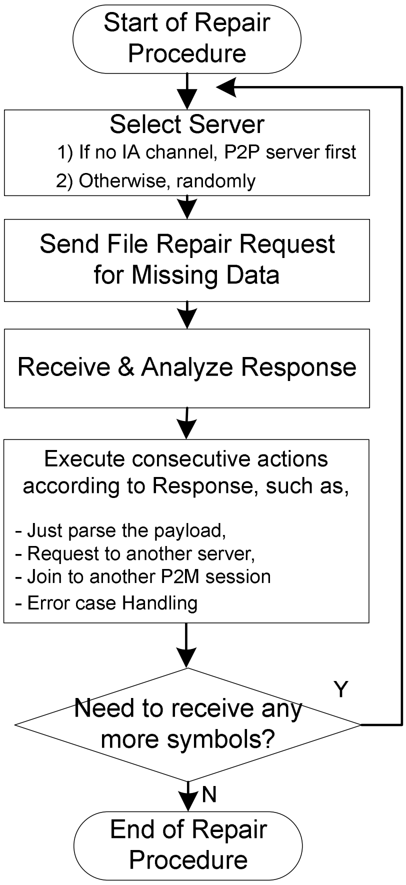
Start of Repair Procedure
Select Server
1) If no IA channel, P2P server first
2) Otherwise, randomly
Send File Repair Request for Missing Data
Receive & Analyze Response
Execute consecutive actions according to Response, such as,
- Just parse the payload,
- Request to another server,
- Join to another P2M session
- Error case Handling
Need to receive any more symbols?
Y
N
End of Repair Procedure

Figure 13: Example of receiver Behaviour

6.2.2 Reception reporting

Reception reporting is a procedure designed to collect statistics about the service or to confirm reception of a file in a file delivery session. In the following clause, an example of a reception reporting procedure is provided. The example starts from the associated delivery procedure description and ends at the server response to the reception reporting request.

6.2.2.1 Request for reception reporting

The service provider may indicate to terminals that it requires them (or a subset of them) to deliver reception reports upon session end. There are three different types of reception reports: reception acknowledgement, statistical reporting for successful reception, and statistical reporting for all content reception. The following example shows an associated delivery procedure description declaring a reception reporting request of type reception

```
<?xml version="1.0" encoding="UTF-8"?>
   <associatedProcedureDescription xmlns="urn:dvb:ipdc:cdp:associatedProcedures:2005"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="urn:dvb:ipdc:cdp:associatedProcedures:2005
      associated-procedure-description.xsd">
   <postReceptionReport
      offsetTime="100"
      randomTimePeriod="1000"
      reportType="rack">
      <serverURI>http://www.provider1.com/ReportService</serverURI>
      <serverURI>http://www.provider2.com/ReportService</serverURI>
      <serverURI>http://www.provider3.com/ReportService</serverURI>
   </postReceptionReport>
</associatedProcedureDescription>
```

6.2.2.2 Reception report

Upon ending the file delivery session, terminals that have an interactive channel have to generate a reception report and send it to one of the designated reception report servers. The reception report server is selected randomly from the list of the reception reporting servers defined in the associated delivery procedure description. The sending time of the reception report is also picked randomly from the interval defined by the offsetTime and the randomTimePeriod.

In the following, an example of the reception report is:

```
<?xml version="1.0" encoding="UTF-8"?>
<receptionReport xmlns="urn:dvb:ipdc:cdp:receptionReportRequest:2005"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="urn:dvb:ipdc:cdp:receptionReportRequest:2005
   receptionReportRequest.xsd">
   <receptionAcknowledgement>
      <fileURI>http://www.filedelivery.com/file1.3gp</fileURI>
      <fileURI>http://www.filedelivery.com/file2.mp3</fileURI>
      <fileURI>http://www.filedelivery.com/file3.mp3</fileURI>
   </receptionAcknowledgement>
</receptionReport>
```

The MIME type in the Content-Type header field is set to:

`'Content-Type: application/vnd.dvb.ipdc.reception-report'.`

The HTTP message of the request may look as follows.

```
POST http://www.provider.com/ReportService2 HTTP/1.1
Content-Type: application/vnd.dvb.ipdc.reception-report
Content-Length: 372

<?xml version="1.0" encoding="UTF-8"?>
<receptionReport xmlns="urn:dvb:ipdc:cdp:receptionReportRequest:2005"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="urn:dvb:ipdc:cdp:receptionReportRequest:2005
   receptionReportRequest.xsd">
   <receptionAcknowledgement>
      <fileURI>http://www.filedelivery.com/file1.3gp"</fileURI>
      <fileURI>http://www.filedelivery.com/file2.mp3"</fileURI>
      <fileURI>http://www.filedelivery.com/file3.mp3"</fileURI>
   </receptionAcknowledgement>
</receptionReport>
```

6.2.2.3 Reception report response

The following is a typical response from the reception report server to the terminal.

```
HTTP/1.1 200 OK
```

6.3 Use and configuration of application layer FEC

This clause analyses and gives examples of the use of application layer FEC. Recommended deployments are also discussed.

6.3.1 General

Application Layer FEC (AL-FEC) is most valuable in cases where the data set to be delivered is to be spread over more than one DVB-H Time Slice Burst. In networks where services are provided based on data sets which are delivered over multiple DVB-H time slice bursts it is strongly recommended that AL-FEC is supported by all terminals.

The data set to be delivered may consist of a single file or multiple files which are required together in order to deliver a service. Data may be spread over more than one time slice burst when the data set to be delivered is larger than a single burst, or if multiple files are interleaved such that each file is spread over multiple bursts.

For data sets which are delivered over multiple DVB-H time slice bursts, in the absence of AL-FEC then efficient file delivery is only possible within the coverage area where the MPE-FEC frame error rate (the fraction of MPE frames which could not be completed recovered using MPE-FEC) is close to zero for terminals supporting MPE-FEC or where the TS error rate is close to zero for terminals not supporting MPE-FEC. Furthermore, reception at any point where the MPE-FEC frame error rate is non-zero (for terminals supporting MPE-FEC) or the TS error rate is non-zero (for terminals not supporting MPE-FEC) requires reception of multiple repetitions of the data.

The AL-FEC operates by providing additional "repair packets" which can be used to reconstruct the data set as soon as a sufficient quantity of data packets has been received, independent of the mix of source and repair packets that have been received. The key difference from the case of simple repetition is that the terminal in general does not receive a duplicate copy of the same packet (in which case the duplicated reception is useless) and thus the transmission is significantly more efficient. For AL-FEC, a sufficient quantity of received packets is usually slightly larger than the size of the data to be delivered: reception of an amount of data 1 % greater than the file size will generally allow recovery of the file in over 99,9 % of cases (see note).

NOTE: For small files, then a single packet may represent more than 1 % of the file and in this case receipt of a number of data packets one greater than the size of the file is sufficient to result in a decoding probability of at least 99,9 %.

File reception progresses at approximately the arrival rate of data packets independent of the reception conditions. For example, file reception in conditions with 5 % packet loss takes only a small fraction longer than reception in conditions with 0 % loss. This is very different from a classical data carousel where data is simply repeated. In that case, reception of duplicate packets happens frequently and does not contribute to reception of the file.

Figure 13a shows an example of the delivery time for a 95 % probability of reception in various channel conditions. Further examples are provide in annex A. Note that cases shown as 150 time slice bursts in this graph are in fact cases where the simulation did not terminate i.e. the file was not successfully delivered at all.

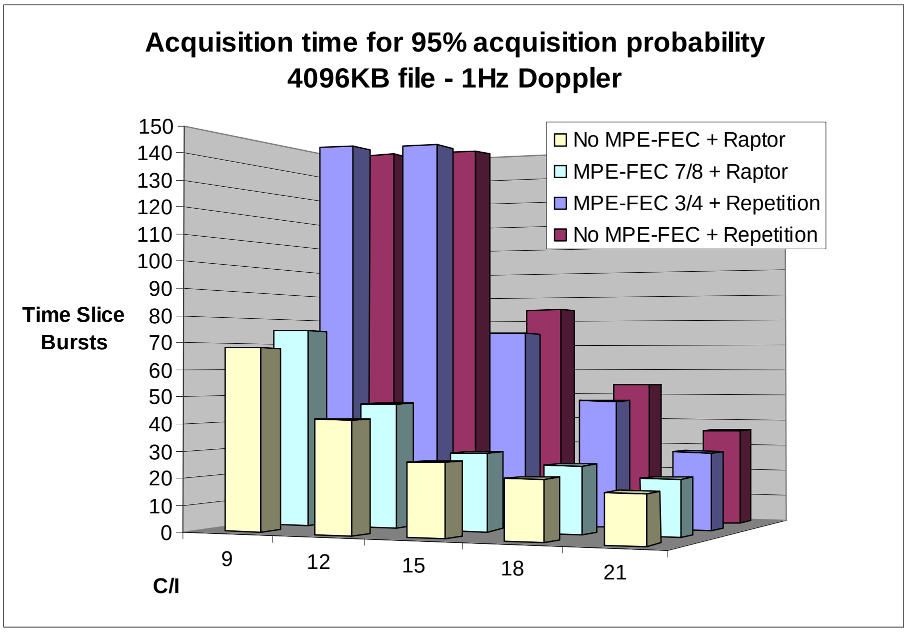

Figure 13a: Example delivery time results for 4 096 KB file

Due to the slow fading characteristics of the DVB channel, then mobile terminals in poor reception conditions will still experience relative extremes of good and bad reception allowing progress to be made in the reception of the file.

6.3.2 Use of MPE-FEC

Where a transmission includes data intended for terminals which do not support the AL-FEC code then the use of MPE-FEC can improve performance for those terminals which support MPE-FEC. However, assuming the total bandwidth is kept constant, then this improvement will be at the expense of performance for the terminals which support AL-FEC decoding.

Recommendations for a transmission which includes data intended for a mixture of terminals, some of which do and some of which do not support MPE-FEC, and some of which do and some of which do not support AL-FEC are complex and need to be considered on an individual basis depending on the fraction and importance of terminals of each of the four possible types.

The relative delivery time for various file sizes at various C/I points using AL-FEC and using MPE-FEC 3/4 plus repetition of data are shown in figure 13b. These figures are based on the simulation assumptions shown in annex A. In this figure, the delivery times, measured in whole time slice bursts, are scaled according to the time required to deliver the file with no FEC protection in error-free conditions. Note that for small files, then only a few time slice bursts are required and so in low loss cases the additional data required by MPE-FEC compared to AL-FEC does not result in any additional bursts (for example, the 512 KB file fits into 3 time slice bursts in both the AL-FEC case and the MPE-FEC case, although more data is sent within those 3 bursts in the MPE-FEC case).

The C/I values shown here include a 9 dB margin for shadow fading and so are offset from the values usually given based only on Raleigh fading. For reference, the average MPE-FEC Frame Error Rates that were observed in the simulations were as follows (noting firstly that due to the shadow fading, the MPE-FEC FER was not constant during each simulation and secondly that the average varied between simulation cases - as shown in the ± values in table 2 below). As can be seen from these values, the simulations cover a C/I range from "within MPE-FEC coverage" (21 dB) to well outside the coverage area generally considered to be supported by MPE-FEC (12 dB). For example in the DVB-H validation exercise the MPE-FEC FER of 5 % was considered to be the border of the coverage.

Table 2: Simulated MPE-FEC FER with different C/I values including shadow fading margin

C/I including shadow fading margin	Observed MPE-FEC FER in simulations
12 dB	68 % ± 8 %
15 dB	30 % ± 7 %
18 dB	10 % ± 2 %
21 dB	0,35 % ± 0,03 %

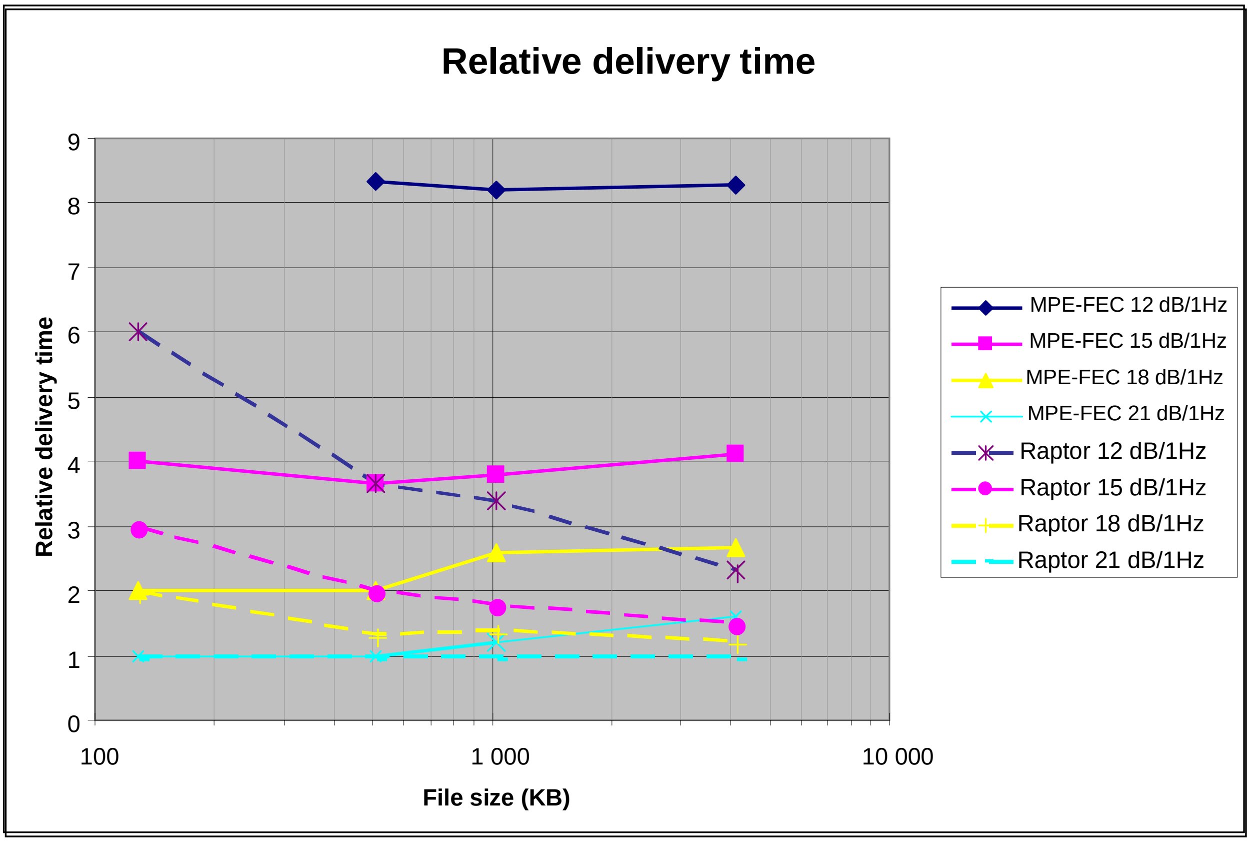

Figure 13b: Relative delivery times with different forms of FEC in different reception conditions

6.3.3 Guidelines for configuration of AL-FEC

This clause provides guidelines for configuration of AL-FEC for the recommended case in which all terminals support AL-FEC decoding. Configuration of AL-FEC should be based on the target quality of service for the file delivery session. There are three metrics which measure the quality of service:

- the transmission time or file reception time;
- the coverage area;
- the channel bandwidth required.

Given any two of these three metrics as fixed parameters, then AL-FEC can be configured to optimize the third. Alternatively, intermediate configurations may be used which provide a balance between the three metrics.

In order to provide service within a given coverage area, transmissions must be configured to achieve a defined quality target at the edge of the coverage area. A quality target for file delivery can be defined in terms of a target delivery time and a target delivery probability (i.e. the probability that a terminal which has received for the entire target delivery time has received the file).

The clauses below describe (a) how the bandwidth required can be derived from given reception conditions and target delivery time and probability (fixed delivery time case) and (b) how the required delivery time can be derived from given reception conditions and available bandwidth (fixed bandwidth case).

The actual transmission duration should be at least equal to the target delivery time in order to ensure that terminals which listen to the whole transmission receive the file with the required probability. The transmission duration may be longer than this target delivery time in order to provide service to terminals which begin listening in the middle of the session.

6.3.3.1 Fixed target delivery time

One metric or target for the performance of the file delivery carousel is the average time required between a terminal joining the session and the terminal having received the file or files for given reception conditions. This is known as the target delivery time. In practice, it is never possible to guarantee reception of a file, so the target delivery time is usually given as the time required to achieve a specified probability (e.g. 99 %) of receiving the file.

The bandwidth required can then be determined by considering the following factors:

- the size of the file or files to be delivered;
- the target delivery time in some given reception conditions.

In practice, terminals will move through different reception conditions whilst receiving a file. However, if AL-FEC is used then progress will be made in the reception of the file whatever the reception conditions, provided some correct data is being received.

Suppose:

- "$p_1 = P_{loss}(SNR1)$" is the average packet loss at the edge of the required coverage area, after MPE-FEC correction if applicable;
- "T_{receiv} " is the target delivery time;
- "N_{source} " is the number of source packets in the file;
- "$\mathrm{P}_{\mathrm{size}}$" is the packet payload size in bytes;
- "F" is a factor to account for overheads, including FLUTE, UDP and IP headers, MPE section headers and TS headers, and MPE-FEC if any, i.e.

$$F_0 = \frac{(P_{size} + H_{FLUTE})}{P_{size}} \times \left(\frac{(P_{size} + H_{FLUTE} + H_{MPE})}{(P_{size} + H_{FLUTE})} + \frac{(1 - R_{MPE-FEC})}{R_{MPE-FEC}} \frac{(N_{rows} + H_{MPE})}{N_{rows}} \right) \times \frac{188}{184} \quad (7)$$

- where:
  - "$R_{MPE\text{-}FEC}$" is the MPE-FEC rate;
  - "N_{row}" is the number of rows in the MPE-FEC table;
  - "H_{FLUTE}" is the IP/UDP/FLUTE header size in bytes (i.e. 44);
  - "H_{MPE}" is the MPE section header size (i.e. 16), plus the TS *pointer-field* size (i.e. 1) which is included in the TS packet containing the start of the MPE section (i.e. 17);
  - For example, for P_{size} = 512 and $R_{MPE\text{-}FEC}$=1 then F_0 = 1,143. For $R_{MPE\text{-}FEC}$=0,75 and N_{rows} = 1 024 then F_0=1,519.

- "P_{target}" is the target reception probability.

Then we can derive the following output parameters,

- *N* is the total number of packets to be sent;
- *B* is the required channel bit-rate in bits/s.

NOTE 1: p_1 may be determined by simulation or direct measurement. In this calculation we assume that MPE-FEC is not used; if MPE-FEC is used and supported by the terminals, then the input parameter p_1 should be based on the loss rate after MPE-FEC and the overhead factor, F_o, should be adjusted accordingly to account for the MPE-FEC overhead.

Firstly, assuming that packet losses are distributed independently (see note 2), then the probability that a given packet has been received is just given by (1-p_1). For successful file reception by a receiver which supports AL-FEC, we can assume that receipt of a total number of distinct packets which is at least 1 % greater than the number of source packets is sufficient (see note 3).

NOTE 2: The assumption that packet losses are distributed independently is not in general true. However, these calculations are presented for illustration and guideline purposes. In practice it is necessary for practical experience of file delivery performance to be used to determine the optimum configuration.

NOTE 3: Successful decoding is sometimes possible when a number of packets which is less than 1 % more than the number of source packets has been received. Receipt of a number of packets which is 1 % greater than the number of source packets results in a probability of successful file decoding of approximately 99,9 % or more. Receipt of a number of packets which is 2 % greater than the number of source packets results in a probability of successful decoding of approximately 99,9999 % or more. This curve is so steep that the above approximation is well within the accuracy of the calculations described here. Note that for small files, then a single packet may represent more than 1 % of the file and in this case receipt of a single such packet is sufficient to result in a decoding probability of approximately 99,9 % and two such packets will result in a decoding probability of 99,9999 %.

The probability that the number of packets received is not less than *m* after *N* packets have been sent is given by the cumulative Binomial distribution as follows:

$$P_{received\ \geq m} = \sum_{i=m}^{i=N} \begin{pmatrix} N \\ i \end{pmatrix} (1 - p_1)^i \, p_1^{\ N-i} \tag{8}$$

And so we must choose *N* to be the least number such that:

$$P_{received \geq (1.01 N_{source})} \geq P_{t\arg et} \tag{9}$$

As a result the bandwidth dedicated to the session is given by:

$$B = \frac{N \times P_{size} \times F_o \times 8}{T_{receive}} \qquad (10)$$

Table 3 provides some examples for different file sizes.

Table 3: Examples for fixed target delivery time with different file sizes (AL-FEC terminals)

File Size	8 192 KByte	1 024 KByte	128 KByte	16 KByte
Source Payload Size, P_{size}	512 bytes	512 bytes	512 bytes	512 bytes
$P_{size} \cdot F_o$	585,5 bytes	585,5 bytes	585,5 bytes	585,5 bytes
Packet loss at edge of coverage (p_1)	25 %	25 %	25 %	25 %
Target reception time - AL-FEC receivers ($T_{receive}$)	256 s	32 s	4 s	0,5 s
Target reception probability (P_{target})	99 %	99 %	99 %	99 %
Transmitted packets (N)	22 268	2 832	371	54
Total Bandwidth	407 kbit/s	415 kbit/s	434 kbit/s	506 kbit/s

The bandwidth requirements above may be reduced by allowing a longer target reception time.

The AL-FEC repair packets should not be repeated until all possible repair symbols have been sent. Since the number of possible repair symbols is large, this should not occur except in the case of very large files and very long transmissions times. However, if the AL-FEC repair packets are repeated then subsequent repetitions after the first should send them in random order.

It should be noted that the assumption of independent random packet loss means that the positive effects of interleaving are not considered in these formula. Thus it is not possible to obtain results for multiple interleaved files in a straightforward fashion.

6.3.3.2 Fixed session bandwidth

In the case that the session bandwidth is fixed, the service provider must determine the required delivery time to achieve the target delivery probability.

The required delivery time is determined by several factors:

- the bandwidth of the channel;
- the size of the file or files to be transmitted;
- the coverage area within which reception is required.

The minimum possible required delivery time is clearly the size of the file or files (plus packetization overheads) divided by the bandwidth.Such a delivery time would provide for delivery only to those terminals in "perfect" coverage for the time. Since such perfect coverage can normally not be guaranteed everywhere within the service area, a particular receiver will experience a certain reception probability.

When a larger coverage area and/or reception probability is required then AL-FEC repair packets should be sent after the initial transmission of the file. The more AL-FEC packets that are sent the larger the coverage area within which the file can be received, and/or the larger the reception probability.

The number of packets that need to be sent to achieve the target delivery probability, N, is calculated as described in clause 6.4.3.2 above. From this, we can determine the delivery time, *D*, which is the length of time that a receiver at the edge of the target coverage area must be tuned in to the session in order to achieve the target file reception probability:

$$D = \frac{N \times P_{size} \times F_o \times 8}{B} \qquad (11)$$

Table 4 provides some examples based on the formula above.

Table 4: Examples for fixed session bandwidth with different file sizes (AL-FEC terminals)

File Size	8 192 KByte	1 024 KByte	128 KByte	16 KByte
Bitrate	512 kbit/s	512 kbit/s	512 kbit/s	512 kbit/s
P_{size}	512 bytes	512 bytes	512 bytes	512 bytes
$P_{size}.F_o$	588 bytes	588 bytes	588 bytes	588 bytes
Packet loss at edge of coverage for AL-FEC terminals (p_1)	25 %	25 %	25 %	25 %
Target reception probability	99 %	99 %	99 %	99 %
Transmitted packets (*N*)	22 268	2 832	371	54
Delivery time	203,7 s	25,9 s	3,4 s	0,5 s

Note that when considering collections of separately protected files, it is not correct simply to look at the combined data size in the table above. For example, a collection of 8 1 MB files would have a longer delivery time than a single 8 Mb file. However, if it is required to achieve the target reception probability for the whole group of files, then it is also not correct to multiply the delivery time associated with the file size by the number of files.

EXAMPLE: 8 times the delivery time for a 1 MB file is 208 s.

After this time the reception probability for each file is 99 % and so the probability of receiving all 8 files is about 96 % - under the assumption of independent random packet loss then a longer time would be required to achieve a 99 % probability for the group of files. In the presence of correlated (bursty) losses, however, the time required will be closer to 8 times the single file delivery time.

6.3.4 Compatibility with non-AL-FEC terminals

In a given deployment it may be necessary to deliver data to both, terminals that implement AL-FEC, and those that do not. However, in this case the coverage benefits and/or bandwidth savings will not be fully realized.

In order to support such compatibility, it is necessary to ensure that the transmitted data includes a correct mix of packets containing source data and packets containing AL-FEC repair data.

The AL-FEC code generates a continuous stream of repair packets, each of which is equally useful to a receiver in the recovery of lost source packets. Providing compatibility with non-AL-FEC capable terminals implies that transmission of these repair packets must be interrupted with re-transmissions of the original data in order to satisfy those terminals without support for AL-FEC decoding. Such retransmitted packets are only useful to receivers in the case that the source packet in question has not already been received.

As a result, the efficiency of file delivery to receivers with the AL-FEC code is compromised by the requirement to provide compatibility for those receivers which do not support the AL-FEC code - the greater the frequency of repetition of the source data within the repair data stream then the greater the impact. Conversely, the lower the frequency of repetition of the source data, the longer the average delivery time to terminals without the AL-FEC decoder.

The benefits of AL-FEC are maximized when it is known that all terminals support AL-FEC decoding and thus the source data need not be repeated until the sequence of repair packets is exhausted.

This consideration motivates the strong recommendation above that all terminals should support AL-FEC decoding for the services considered here.

6.3.4.1 Fixed target delivery time

The bandwidth required to achieve a fixed target delivery time for a mixed terminal population can then be determined by considering the following factors:

- the size of the file or files to be delivered;
- the target delivery time in some given reception conditions for receivers with AL-FEC decoding support;
- the target delivery time in some given reception conditions for receivers without AL-FEC decoding support.

Suppose:

- "$p1 = P_{loss}(SNR1)$" is the average packet loss at the edge of the required coverage area for terminals without AL-FEC support, after MPE-FEC correction if applicable;
- "$p2 = P_{loss}(SNR2)$" is the average packet loss at the edge of the required coverage area for terminals with AL-FEC support, after MPE-FEC correction if applicable;
- "$T_{receive}$" is the target delivery time;
- "N_{source}" is the number of source packets in the file;
- "P_{size}" is the packet payload size, in bytes;
- "F_o" is a factor to account for overheads, including FLUTE, UDP and IP headers, MPE section headers and TS headers, and MPE-FEC if any;
- "P_{target}" is the target reception probability.

Then we can derive the following output parameters:

- "N_{repeat}" is the number of times the source data must be repeated within the target delivery time in order to provide service to the non-AL-FEC receivers;
- "N_{repair}" is the number of repair packets that must be sent during the target delivery time in order to provide service to the AL-FEC receivers;
- "B_1" is the required channel bit-rate in bits/s dedicated to transmission of source data;
- "B_2" is the required channel bit-rate in bits/s dedicated to transmission of repair data;
- "B" is the required channel bit-rate in bits/s.

NOTE 1: p_1 and p_2 may be determined by simulation or direct measurement. In this example, for simplicity, we assume that the target reception time is the same for both kinds of receiver. The implications of different target reception times are discussed below. In this calculation we assume that MPE-FEC is not used; if MPE-FEC is used and supported by the terminals, then the input parameters p_1 and p_2 should be based on the loss rate after MPE-FEC and the overhead factor, F_o,should be adjusted accordingly to account for the MPE-FEC overhead.

Firstly, assuming that the source packets are distributed within the transmission such that their losses are independent (see note 2), then the probability that a given source packet has been received after N_{repeat} repetitions by a terminal in conditions *SNR1* is given by:

$$1 - p_1^{N_{repeat}} \qquad (12)$$

NOTE 2: The assumption that packet losses are distributed independently is not in general true. However, these calculations are presented for illustration and guideline purposes. In practice it is necessary for practical experience of file delivery performance to be used to determine the optimum configuration.

Thus the required number of repetitions of the source data, N_{repeat}, can be determined by choosing N_{repeat} to be the lowest number such that:

$$(1 - p_1^{N_{repeat}})^{N_{source}} \geq P_{target} \qquad (13)$$

In order to ensure that this target reception is achieved within the target time $T_{receive}$, then the bandwidth dedicated to repetition of source data is given by:

$$B_1 = \frac{N_{source} \times N_{repeat} \times P_{size} \times 8}{T_{receive}} \tag{14}$$

Next, for the AL-FEC receivers in the worse conditions, *SNR2*, then the probability that a given source packet has been received directly is given by:

$$1 - p_2^{N_{repeat}} \tag{15}$$

The probability that exactly *n* source packets have been received is given by the Binomial distribution as follows:

$$P_{source\_received=n} = \binom{N_{source}}{n} (1 - p_2^{N_{repeat}})^n \, p_2^{N_{repeat} \cdot (N_{source} - n)} \tag{16}$$

For successful reception by receiver which supports AL-FEC, we can assume that receipt of a total number of distinct packets which is 1 % greater than the number of source packets is sufficient. The probability that the number of repair packets received is not less than *m* is given by the cumulative Binomial distribution as follows:

$$P_{repair\_received \geq m} = \sum_{i=m}^{i=N_{repair}} \binom{N_{repair}}{i} (1 - p_2)^i \, p_2^{N_{repair} - i} \tag{17}$$

And so we must choose N_{repair} to be the least number such that:

$$\sum_{i=0}^{i=N_{source}} P_{source\_received=i} \times P_{repair\_received \geq (1.01 N_{source} - i)} \geq P_{t\arg et} \tag{18}$$

As a result the bandwidth dedicated to repair packets is given by:

$$B_2 = \frac{N_{repair} \times P_{size} \times F_o \times 8}{T_{receive}} \tag{19}$$

And so the total bandwidth $B = B_1 + B_2$.

Table 5 provides some examples for a 4 MB file. The first column assumes that the non-AL-FEC receivers are in perfect conditions (0 % loss) and so is identical to the case where non-AL-FEC receivers are not considered described in clause 6.3.3.

Table 5: Examples for fixed target delivery time with different file sizes (AL-FEC and non-AL-FEC terminals)

File Size	4 096 KByte	4 096 KByte	4 096 KByte	4 096 KByte
P_{size}	512 bytes	512 bytes	512 bytes	512 bytes
$P_{size}.F_o$	585,5 bytes	585,5 bytes	585,5 bytes	585,5 bytes
Packet loss at edge of coverage for non-AL-FEC receivers (p_1)	0 %	0,1 %	1 %	5 %
Packet loss at edge of coverage for AL-FEC receiver (p_2)	25 %	25 %	25 %	25 %
Target reception time - ($T_{receive}$)	120 s	120 s	120 s	120 s
Target reception probability (P_{target})	99 %	99 %	99 %	99 %
Source repetitions (N_{repeat})	1	2	3	5
Source packets sent ($N_{source}.N_{repeat}$)	8 192	16 384	24 576	40 960
Repair packets sent (N_{repair})	3 008	880	352	192
Source bandwidth (B_{source})	321 kbit/s	642 kbit/s	962 kbit/s	1,6 Mbit/s
Repair bandwidth (B_{repair})	118 kbit/s	34 kbit/s	14 kbit/s	8 kbit/s
Total Bandwidth	439 kbit/s	676 kbit/s	97 7kbit/s	1,6 Mbit/s

Figure 14 plots the values above and similar values for different file sizes. It should be noted that these values are subject to a quantization effect because they are constrained to a whole number of repetitions. As a result, the actual reception probability achieved, although always above 99 %, differs in different cases.

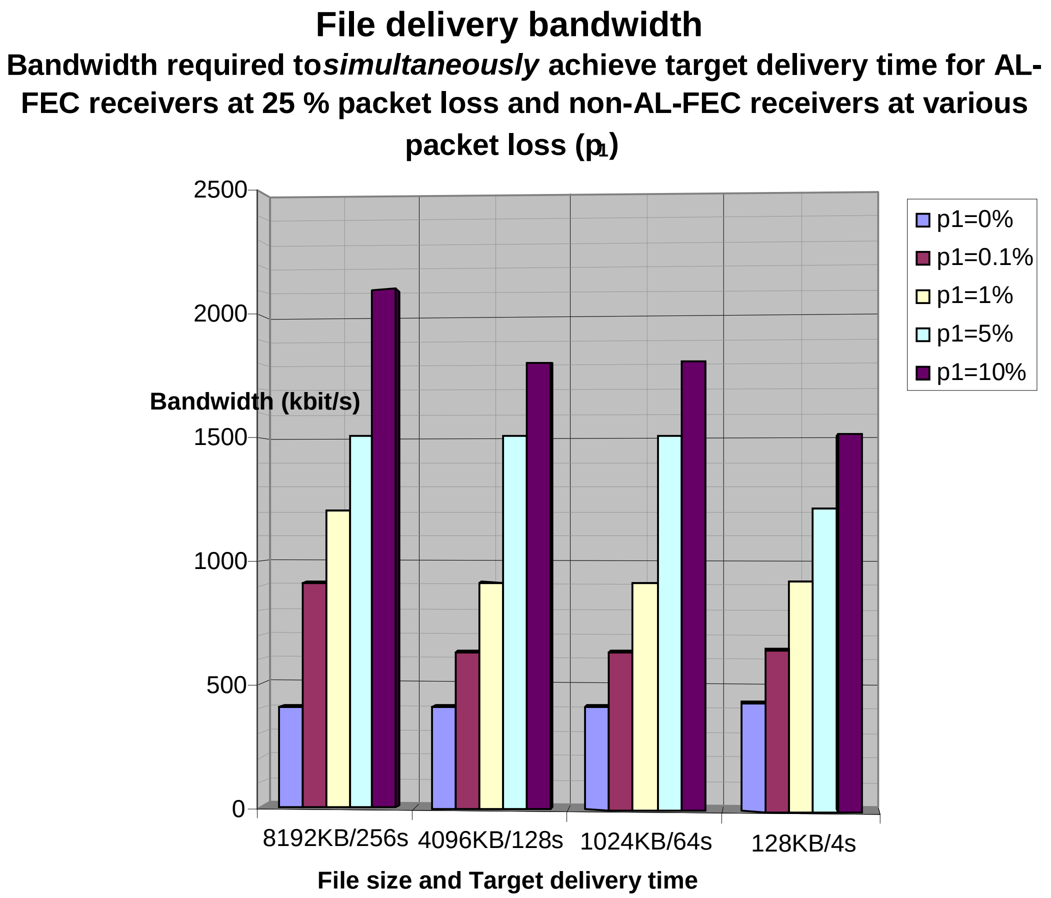

Figure 14: Bandwidth required to simultaneously achieve target delivery time for both AL-FEC and non-AL-FEC terminals

It should be noted that the bandwidth requirements in the case that support is required for non-AL-FEC terminals are very significantly higher than in the recommended case where all terminals support AL-FEC. In fact in these cases, most of the bandwidth required is being used to provide service to terminals without AL-FEC support even though they are suffering relatively modest losses.

In the above examples, different coverage areas were assumed for AL-FEC and non-AL-FEC terminals and the system engineered to achieve the same delivery time for both kinds of terminals at the edge of their respective coverage areas. Alternatively, in the case of long-lasting file carousels, a single coverage area could be targeted and the delivery time for the different types of terminals would then be very different. In practice, operators may want to configure the system based on a compromise between these two extremes, providing a short delivery time over a wide coverage area for AL-FEC terminals and a longer delivery time over a smaller coverage area for non-AL-FEC terminals.

NOTE 3: "Long lasting" is a rather vague characterization: in fact the number of rounds the carousel should stay alive depends on the size of the largest file and the packet loss and may be calculated from the formulae for N_{repeat} given above.

6.3.4.2 Fixed session bandwidth

In the case that the bandwidth is fixed, then the required transmission duration for a mixed population of terminals, is determined as follows.

The duration of the file delivery session is determined by several factors:

- the bandwidth of the channel;
- the size of the file or files to be transmitted;
- the coverage area within which reception is required for terminals with AL-FEC support;

- the coverage area within which reception is required for terminals without AL-FEC support.

As above, suppose:

- "$p1 = P_{loss}(SNR1)$" is the average packet loss at the edge of the required coverage area for terminals without AL-FEC support, after MPE-FEC correction if applicable;
- "$p2 = P_{loss}(SNR2)$" is the average packet loss at the edge of the required coverage area for terminals with AL-FEC support, after MPE-FEC correction if applicable;
- "N_{source}" is the number of source packets in the file;
- "P_{size}" is the payload packet size, in bytes;
- "F_o" is a factor to account for overheads, including FLUTE, UDP and IP headers, MPE section headers and TS headers, and MPE-FEC if any;
- "B" is the channel bit-rate in bits/s;
- "P_{target}" is the target reception probability.

Then we can derive the following output parameters:

- "D_{source}" is the delivery time which must be devoted to source data;
- "D_{repair}" is the delivery time which must be devoted to repair data;
- "D" is the total delivery time.

The probability that a non-FEC receiver receives a given source packet and the number of repetitions required (N_{repeat}) is given by the formulae above.

From this, we can determine the delivery time which must be devoted to source data, D_{source}. as follows:

$$D_{source} = \frac{N_{source} \times N_{rep} \times P_{size} \times F_o \times 8}{B} \qquad (20)$$

As above we can further calculate the number of repair packets which must be sent N_{repair}, which can then be used to derive the delivery time which must be devoted to repair data:

$$D_{repair} = \frac{N_{repair} \times P_{size} \times F_o \times 8}{B} \qquad (21)$$

So, finally, the delivery time is given by $D_{session} = D_{source} + D_{repair}$.

The order of sending of packets is also important. In order to provide optimum service to the receivers without AL-FEC support, then repetitions of the source packets should be spaced as far apart as possible. However, since the DVB channel exhibits slow fading, with potentially long periods of correct reception, then all the source packets of each repetition should be grouped together. Further, since some receivers may experience "perfect" reception, it is desirable to send a complete set of source packets at the start of the transmission. If $N_{repeat} > 1$, then it is recommended that this set of source packets should be followed by a group of repair packets of size N_{repair_group}, where:

$$N_{repair\_group} = \frac{N_{repair}}{N_{repeat} - 1} \qquad (22)$$

This sequence of source packets followed by N_{repair_group} repair packets is then repeated until all repair packets have been sent. A final copy of the source packets is then sent to complete the session. The sending sequence for a session with N_{repeat}=3 is shown in figure 15.

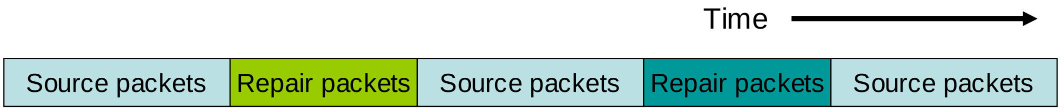

Figure 15: Sending order for N_{repeat} = 3

Note that repair packets should never be repeated except in the case that the session duration is longer than the time required to send all possible AL-FEC symbols. In that case, then the repair packets should be repeated except that in transmissions subsequent to the first the order of transmission of repair packets should be randomized.

Table 6 provides some examples based on the formulae above. Again, the first column assumes the non-AL-FEC receivers are in perfect conditions and so duplicates the calculation of clause 6.3.3.

Table 6: Examples for fixed session bandwidth with different file sizes (AL-FEC and non-AL-FEC terminals)

File Size	4 096 KByte	4 096 KByte	4 096 KByte	4 096 KByte
Bitrate	512 kbit/s	512 kbit/s	512 kbit/s	512 kbit/s
Source Payload Size, P_{size}	512 bytes	512 bytes	512 bytes	512 bytes
$P_{size}.F_o$	588 bytes	588 bytes	588 bytes	588 bytes
Packet loss at edge of coverage for non-AL-FEC terminals (p_1)	0 %	0,1 %	1 %	5 %
Packet loss at edge of coverage for AL-FEC terminals (p_2)	25 %	25 %	25 %	25 %
Target reception probability	99 %	99 %	99 %	99 %
Source repetitions (N_{repeat})	1	2	3	5
Source packets sent ($N_{source}.N_{repeat}$)	8 192	16 384	24 576	40 960
Repair packets sent (N_{repair})	3 008	880	352	192
Transmission duration	102 s	158 s	228 s	404 s

It should be noted that the bandwidth requirements in the case that support is required for non-AL-FEC terminals are very significantly higher than in the recommended case where all terminals support AL-FEC. In fact in these cases, most of the bandwidth required is being used to provide service to terminals without AL-FEC support even though they are suffering relatively modest losses.

6.3.5 Interleaving of multiple files

When multiple files are to be sent over a single DVB-H channel, then the data packets for these files may be sent either sequentially or in an interleaved fashion.

In the sequential approach, a number of packets are sent for the first file, followed by a number of packets for the second file and so on. In general, the number of packets sent for each file is at least equal to the number of source packets in the file. If the number of packets sent is equal to the number of source packets and AL-FEC is not applied, then this approach corresponds to the traditional carousel approach with the same cycle time for every file. In the following we refer to a "carousel cycle" as the time taken to send all the source packets of all the files. It should be noted that in practice, some files may be repeated more often than others.

In the case of files that are very small with respect to the size of the DVB-H time slice burst, this approach generally results in sufficient data to receive the file appearing within a single DVB-H time slice burst.

In the interleaved approach, packets from the multiple files are interleaved such that one packet is sent for each file in turn. Even in the case if files that are small with respect to the size of a DVB-H time slice burst, this approach may result in packets from a single file being distributed over multiple bursts.

In the case that each of the multiple files are useful to the receiver independently, sequential sending has the advantage that a file may be received by the receiver very rapidly once it has tuned in to the channel (e.g. if the required file happens to be next to be transmitted). The reception time for a randomly chosen file in perfect reception conditions is anything between zero and a full carousel cycle, dependent on the moment when the receiver tunes in. In the interleaved approach, always a full carousel cycle is required to receive any one of the files.

However, in less than perfect reception conditions, with sequential transmission, there is still a chance that a given file will be recovered after its first appearance on the carousel (which will take on average half a cycle). However, when this first appearance is received with errors, at least a further full carousel cycle is required to recover the file. In the case of interleaved transmission, whilst at least full initial carousel cycle is always required, the additional time required if reception has not been successful after one cycle is approximately proportional to the average error rate experienced during that cycle and in general considerable less than in the carousel case.

In the case that all files are required to deliver the service to the user, then in perfect reception conditions sequential and interleaved transmission are equivalent - all files are received after one carousel cycle. However, in less than perfect conditions, sequential transmission will almost always result in at least 2 carousel cycles being required to recover all files whereas interleaved transmission may allow the files to be delivered much faster.

Therefore, it is recommended that both interleaving and AL-FEC are used whenever receipt of all files in the set is required or desirable to deliver the service to the user. In the case of multiple independent files, then sequential transmission provides for shorter average reception times in good conditions at a cost of longer average reception times in less good conditions as compared with interleaving.

6.3.6 Use of sub-blocking feature

The AL-FEC code partitions the file to be transmitted into source blocks. Each source block may be further partitioned into sub-blocks. Partitioning into source blocks is required to ensure that the symbol size is less than or equal to the packet size (so that packets only contain whole symbols) and to respect the limit on the total number of symbols. Each source block is encoded and transmitted independently. Source blocks may be very large, for example with 1 024 byte packets, the maximum source block size is 8 Mb.

Partitioning source blocks into sub-blocks may be performed by either the sender or the receiver to support decoding within limited working memory at the receiver. Specifically, each sub-block is small enough to be decoded within working memory whilst remaining sub-blocks are kept in secondary storage (for example flash memory). Partitioning into sub-blocks has no effect on the error correction performance of the code.

Partitioning into sub-blocks may be performed by either the transmitter or the receiver or both. When performed by the transmitter, then each sub-block consists of a contiguous portion of the source block. Partitioning into sub-blocks by the transmitter affects the contents of both source and repair packets and is signalled by setting the parameter ***N*** to the number of sub-blocks per source block. If N=1 then no partitioning is performed and each source block consists of a single sub-block.

When sub-block partitioning is performed by the transmitter, then each packet contains data from every sub-block of the source block. Received data may be stored by the terminal into secondary storage, and then read back one sub-block at a time to perform decoding.

If the sub-block size is too large to be decoded in working memory then the receiver may need to perform its own sub-block partitioning in order to perform decoding. In this case, each sub-block consists of the concatenation of one sub-symbol from each symbol of the block i.e. the sub-block does not consist of a contiguous portion of the source block and thus writing back decoded sub-blocks to secondary storage may be less efficient.

It is not required for transmitters to partition source blocks. In particular, when transmissions are configured for receipt by a mixed population of terminals including terminals which do not support AL-FEC decoding, then transmitters should not perform such partitioning since it is not required to be supported by receivers that do not support AL-FEC decoding. It is therefore important that receivers that support AL-FEC do support receipt of transmissions in which sub-block partitioning has not been performed by the transmitter.

6.4 Delivery of web pages (webcasting)

Webcasting applications are expected to be one of the significant traffic and revenue generators in IPDC. In webcasting, the terminal receives a web site content over the broadcast channel. The webcasting service may operate in a push-mode, where it sends up to date web sites to the receivers without a specific request. The user might have subscribed (locally by joining the multicast group) for such a service a-priori. Figure 16 depicts the use case for webcasting services.

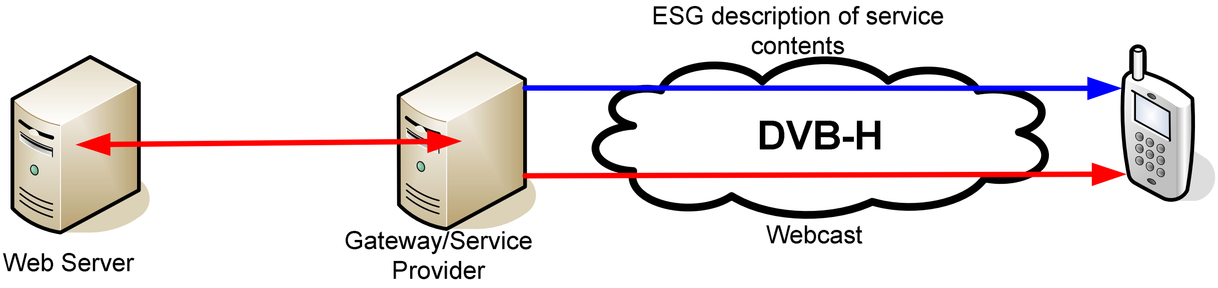

Figure 16: Webcasting in push mode

The terminal detects the webcasting service from the ESG. The terminal discovers the entry point for a particular set of files from the ScheduledEvent fragment (<ContentLocation>).

The server should deliver all the necessary files that build the web site over the FLUTE session. It should also use the grouping mechanism in the FDT to indicate to the receivers that a set of files are related and need to be downloaded together as an entity. It should also deliver the content as entry point faster than the other files, in order to allow for ashort enough access time in case no pre-cashing is done by the terminal.

The FLUTE receiver should download the file with the corresponding URL and all other files that belong to the same group or groups. It is recommended that all files of a group are declared in the same FDT Instance in the FLUTE session. It should be noted that using the original web site content URL as URI in the FLUTE FDT makes implementation of webcasting a lot easier for all the actors in the webcasting chain (content generation, webcasting gateway, terminal). This is especially true when the original web site is partially webcasted, with content remaining available only over the interaction channel.

The FLUTE receiver may automatically download and cache all, or a subset of all, files delivered over the FLUTE session. It may alternatively start download only when the first file from a given site is requested.

A conceptual model for terminal behaviour is illustrated in figure 17. The browser is modelled having a module rendering WEB pages ("Browser Rendering"), a module for handling requests for WEB pages ("Browser Request Handler") and a module for caching WEB pages ("Browser Cache"). The Browser Request Handler forwards requests to a WebCast Proxy which retrieves the requested content either from the FLUTE client or the Interactive channel.

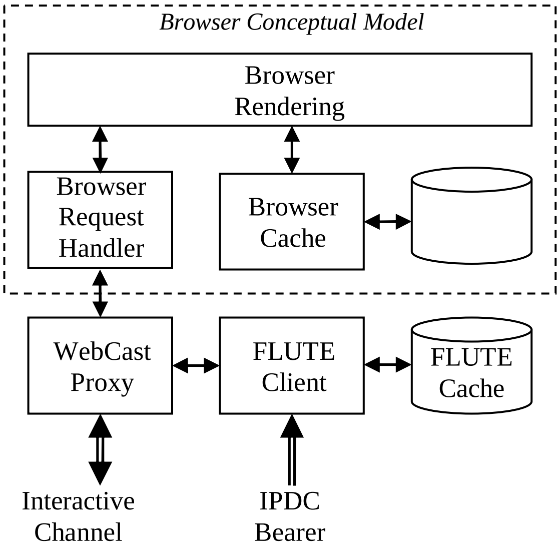

Figure 17: Conceptual model for a FLUTE-aware WEB browser

When navigating a Webcast site the Browser Request Handler is supposed to first look at the Browser Cache. If this action is not successful the Browser Request Handler is assumed to make the request over the interactive channel. In a Webcast - aware implementation a WebCast Proxy would intercept the request in order to check for files corresponding to the required URL(s) in the FLUTE Cache. In the case the needed file(s) cannot be found in the FLUTE Cache neither in the FLUTE session the WebCast Proxy would retrieve the files over the interactive channel. The latter assumes that the terminal is capable of data connection over an interactive channel.

Cache management is browser-specific, however is expected that the Browser Cache will discard files - in particular web pages - that have expired.

The simplest way to describe the content of a web site is to declare all the resources within the same group, as shown in the example figure 18. In this example the web site is made of one root web page and two sub-pages. Each web page is composed of a common script file, a specific script file and media resources.

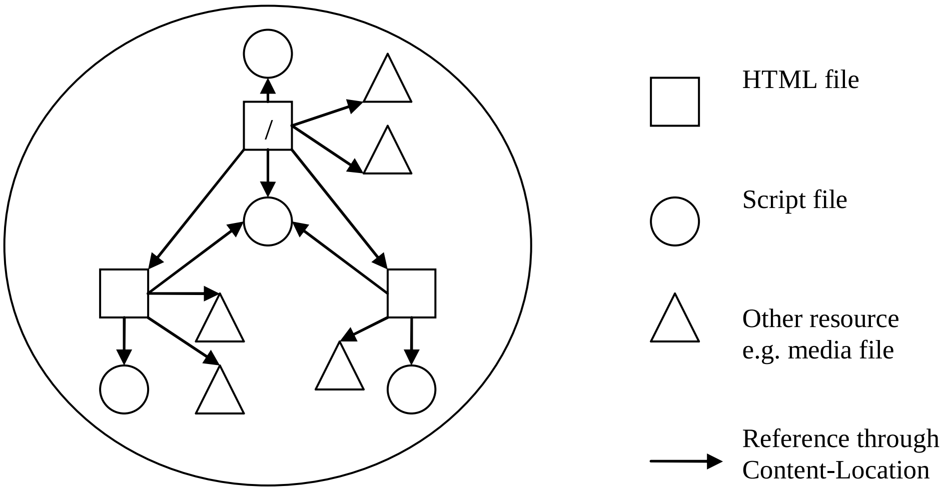
/
HTML file
Script file
Other resource e.g. media file
Reference through Content-Location

Figure 18: Description using one group in the FDT

A more optimized way is to use the grouping mechanism to declare resource per web pages and/or type, as shown in the example figure 19, which deals with the same web site as above. Here, the terminal gets a tree-view of the web site. This allows the terminal to set up an efficient download strategy based on how the groups reference each other in the FDT (e.g. through HTML files).

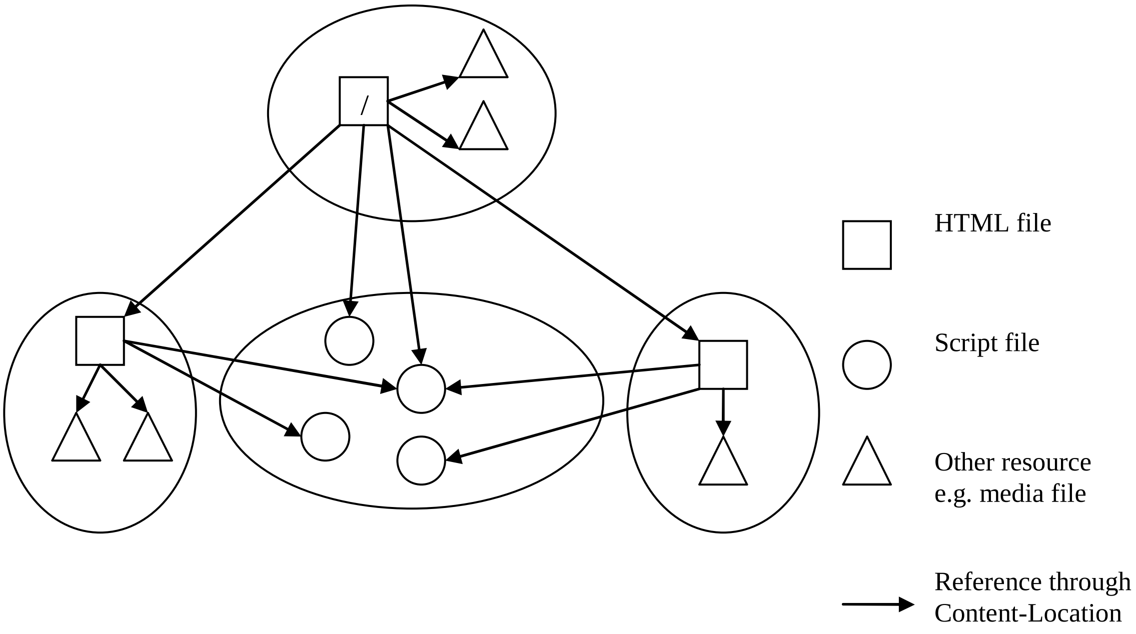

Figure 19: Example description using several groups in the FDT

6.5 FLUTE sessions and memory management

Memory is a limited resource on terminals and must be shared between multiple applications. Participation in a file delivery session can consume a significant amount of memory, depending on the size and number of objects that are received. A terminal has several ways to participate in a FLUTE session, determine the memory allocated for the FLUTE session, and manage it.

6.5.1 Processing of the FDT

Upon joining the delivery session, the terminal may choose to wait for an FDT instance. When the terminal has received and processed one or more FDT instances, the number of files that are available and their respective size are known. The FDT also provides file information (e.g. MIME) that allows the terminal to apply its own handling policy (e.g. keep a file, hand it to the proper application, discard it when it gets too old or when available memory is running low, and so on). The terminal should allocate enough memory to store all the files that are available in the session, although this might not always be possible. In the latter case, the terminal must chose which Transport Objects to retrieve first (e.g. those that can be immediately consumed).

6.5.2 Direct download with delayed FDT

Sometimes an FDT instance might not be immediately available when the terminal joins the FLUTE session. Additionally, the terminal might want to start retrieving transport objects immediately in order to save time, even though no FDT instance is available. To this end, the terminal starts fetching LCT source blocks and groups them according to their TOI. However, one or more FDT instances is needed to identify the length of transported objects and thus validate the received objects. Depending on the available memory, the terminal should set a limit of maximum downloadable material before an FDT instance is actually received and processed, otherwise the terminal may experience memory shortage. The FileDownloadComponentCharacteristic of the acquisition fragment may provide information about the required storage capacity for retrieving the files of the session. Based on this information, the terminal may reserve the necessary memory space for receiving all the files of the session prior to joining the session. When the limit is reached and the FDT is still unavailable and to preserve storage, the terminal should be able discard some of the received Transport Objects or to interrupt the download session till FDT reception and processing.

6.5.3 Immediate consumption of received files

In some cases (e.g. ESG delivery session) files can be consumed immediately after reception. Therefore, the amount of memory that is necessary to participate in the session is lower than for other session types, limited to the maximum size of one object. In most use cases, the terminal is expected to properly handle such FLUTE sessions.

6.5.4 Delayed consumption of received files

In some cases(e.g. Webcasting) files can be consumed only some time after reception. A given amount of memory needs to be allocated for the terminal to store the files while they are waiting for being consumed by the application. The FileDownloadComponentCharacteristic of the acquisition fragment may provide information about the required storage capacity for retrieving the files of the session. Based on this information, the terminal may reserve the necessary memory space for receiving all the files of the session prior to joining the session.

6.5.5 Processing of a new FDT instance

When the FDT is updated, a new set of Transport Objects is signalled to the terminal. There are several situations with impact towards memory management:

- When a new version of a file is available, the older one can be discarded and the corresponding storage capacity freed.
- When the total size of downloadable material decreases, the terminal can allocate spare memory to other tasks (e.g. another delivery session).

When the total size of downloadable material increases, the terminal must consider allocating more memory to the session or revise its download strategy.

6.6 FDT Instance ID wraparound problem

In order to avoid the FDT Instance ID wraparound problem, the Expires attribute of a given delivered FDT Instance should be lower than the time the server will take to create 2^{19} (2^{19} = 524 288) new FDT Instances after this given FDT Instance (i.e. ~6 days in case FDT Instance is updated once a second).

In this case IDs of two unexpired FDT Instances can be compared using 20 bit signed arithmetic. FDT Instance with ID A is newer than FDT Instance with ID B, if A - B > 0.

6.7 FLUTE sessions and Caching

Different applications may be realized based on one or more of delivery session types as defined in clause 6.2 of CDP [1]. The FLUTE receiver may for instance be instructed to operate in one of the following modes:

- Receive all files in a file delivery session.
- Keep receiving the most actual version of a specific file or of all files in the carousel.
- Receive a file or a set of files that have been requested based on their content location.
- Receive files that belong to a file grouping of a requested file.
- Receive files according to some specific characteristics (e.g. mime type).

It is in general not possible to predict the exact delivery time of a specific file. Furthermore, in the cases where the application does not know the exact file(s) to be received, the FLUTE receiver has to store or forward all received files from the FLUTE session (possibly after applying some basic filtering) to the application. Hence, the request/response interaction between applications and the FLUTE receiver typically occurs in an asynchronous manner. The FLUTE receiver is then required to cache received files to make them available for the application upon request.

Furthermore, a specific file may be of use for more than a single application at the receiver. If the file is not cached, consecutive but fairly distanced requests for that file may result in extensive delays due to the need of repeated file retrieval from the FLUTE session.

On the other hand, storage space on mobile terminals may be limited and would not allow for permanent storage of all files received from different file delivery sessions.

The server can provide specific caching directive to the terminal for efficient memory management and for convenient use of content by application. The caching directive could be for individual TO, group of TOs based on FDT instance or a combination of these. When there is such directive on FDT instance level and if there is differing TO level directive, then TO level directive will supersede the FDT instance level directive.

There are four different caching directives defined: no-cache, max-stale caching, caching with a specific expiry time and no-caching directive.

6.7.1 No-cache

The server directs the FLUTE receiver not to cache the TO or group of TOs. The server uses this directive to indicate to the FLUTE receiver that the content has a very short lifetime and should be consumed by the application immediately. It may also indicate that the content would be obsolete if not consumed immediately. The server may use this directive for content that is frequently updated.

One example can be the event schedule (Program guide) update. To indicate the update of a schedule event, notification message can be sent with a file in a FLUTE session. Once, the event schedule is updated the file content is not required and there is no need to cache the content for future use.

6.7.2 Max-stale

The server directs the FLUTE receiver to cache the TO or group of it indefinitely. Max-stale cache directive is used by the server when the file content in the FLUTE session is static in nature and not time dependant.

The information on URLs or addresses (including postal addresses) required for specific applications is an example of max-stale caching directive. In this, file containing a group of web address or postal addresses sent in a FLUTE session would be cached by the FLUTE receiver to be used by the application for locating and retrieving information. Here, the content is not expected to change.

6.7.3 Expires

The server also directs the FLUTE receiver to retain the TO or group of TO until the expiry of the timer as specified. Here, the server determine that content is not valid beyond a specified time. This may be due to such as schedule update that is planned or the repeated usage of contents with limited time.

The weather forecast or replay scene of big sports game may be the examples. Here, the server generally plans to provide an updated weather forecast information periodically and the replay scene of big sports game is expected to be repeated several times before the game is over. Therefore, it will be useful if the file is cached in FLUTE receiver and repeatedly used by applications.

6.7.4 No-caching-directive

The server directs the terminal to manage TO as it wants/can based on any relevant policy it may have implemented.

6.7.5 Default caching directives

6.7.5.1 Absence of caching directive

This is the default situation when the server provides no directive. When there is no directive provided by the server, the memory management and caching is decided by the terminal depending of the TO type and memory available. This is equivalent to the case where a "no-caching-directive" is provided.

6.7.5.2 Caching directive at FDT level

If there is no explicit directive at TO level, then directive at the FDT instance level will be adhered to.

6.7.6 Cache Directive example

The following XML document shows an example of an FDT instance including caching directives for some of the files.

The example also illustrates how an individual file directive will override the FDT instance directive along with the different cache directives.

In this example, cache directive for files carried within the FDT instance is "No-cache". The FDT instance consists of three files; soap_opera_start_time.xml; street_house_postal_codes.xml; and great_goal.mp4.

The caching directive for the FDT instance and Files are as follows:

- FDT instance – "No-cache".
- soap_opera_start_time.xml – no explicit caching directive.
- street_house_postal_codes.xml – "Max-Stale".
- great_goal.mp4 – "Expires".

As result of the override rule, even the FDT instance indicates not to cache all three files, street_house_postal_codes.xml will be cached in for indefinite period, until the power is cycled as directed in the file element, great_goal.mp4 will be retained until the expiry time (XXXYYYY) as directed in the file element, whereas soap_opera_start_time.xml will take the FDT instance directive of "No-cache" even though there is no specific directive to the file element.

```
<?xml version="1.0" encoding="UTF-8"?>
<FDT-Instance
    xmlns="urn:dvb:ipdc:cdp:flute:fdt:2008"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    FEC-OTI-FEC-Encoding-ID="0"
    FEC-OTI-Encoding-Symbol-Length="512"
    Expires="341129600">
    <File
        Content-Type="xml/text"
        Content-Length="3760543"
        TOI="12"
        FEC-OTI-Scheme-Specific-Info="AAEBBA=="
        Content-Location="http://www.example.com/soap_opera_start_time.xml">
    </File>
    <File
        Content-Type="xml/text"
        Content-Length="2934"
        TOI="13"
        Content-Location="http://www.example.com/street_house_postal_codes.xml">
        <Cache-Control>
            <max-stale>true</max-stale>
        </Cache-Control>
    </File>
    <File
        Content-Type="video/mp4"
        Content-Length="21561934"
        TOI="13"
        Content-Location="http://www.example.com/great_goal.mp4">
        <Cache-Control>
            <Expires>347183452</Expires>
        </Cache-Control>
    </File>
    <Cache-Control>
        <no-cache>true</no-cache>
    </Cache-Control>
</FDT-Instance>
```

7 General timing issues

Timing synchronization is necessary for several DVB services to function correctly on the terminal. Among those are FLUTE and RTP sessions. In FLUTE, time synchronization is required as explained in clause 6.1.3. With both FLUTE and RTP, the sender and the receivers need to be synchronized to properly detect start and end times of sessions. Both sender and receivers use UTC in normal operations. Correct synchronization is also required for timing parameters that are present in the ESG.

In order to cope with those various timing issues, it is recommended that the terminal maintains an internal reference clock for DVB services. This reference clock can feed from the Time and Date Table (TDT) as specified in [9]. When the TOT is available, the terminal can maintain a time offset reference thanks to the local_time_offset_descriptor as specified in clause 6.2.18 of EN 300 468 [9].

Annex A (normative): AL-FEC Simulation results

Figures A.1 and A.2 present simulation results for delivery of 1 MB and 4 MB files over DVB-H channels. The DVB-T mode and channel models used to generate the error traces are shown in table A.1.

Table A.1: DVB-T/H mode and channel models

DVB-T Mode	8 k, 16 QAM, CR = 1/2, GI = 1/8
DVB-T TS bit rate	11 Mbit/s
Rayleigh fading	TU6@1Hz, TU6@80Hz
Shadow fading	Lognormal with σ=5,5 dB, correlation distance 20 m
RF Frequency	602 MHz
C/I (note 2)	80 Hz: 15 dB to 27 dB (inclusive) in 3 dB increments 1 Hz: 9 dB to 21 dB (inclusive) in 3 dB increments
Doppler Frequencies	1 Hz, 80 Hz
Length of traces	At least 10 times time taken to pass one correlation distance (i.e. 400 s for 1 Hz Doppler)
NOTE 1: 1 Hz Doppler corresponds to a speed of 0,5 m/s at 600 MHz. It therefore takes 40 s to pass one full correlation distance. NOTE 2: These C/I values include a margin to account for lognormal fading (~9 dB) and so are on a different scale from that more commonly used.	

A service bit-rate of 200 kbit/s was simulated using time slice bursts of 1 099 ms at a peak Elementary Stream rate of 2 Mbit/s and with a frequency of 11 seconds. Data was sent using the FLUTE protocol with 512 bytes of payload data per IP packet.

The figures show the number of time slice bursts required to achieve a 95 % probability of file reception with different FEC configurations and channel conditions. Note that simulations were limited to 50 time slice bursts for the 1 MB file and 150 for the 4 Mb file and thus the cases which show as 50 or 150 bursts respectively represent cases in which the file delivery did not achieve the required delivery probability in that time. These cases are denoted "n/a" in the tabulated results below.

The FEC configurations considered were:

- "No MPE-FEC + repetition" - the file was simply repeated as in a traditional carousel.
- "MPE-FEC ¾ + Repetition" - MPE-FEC was enabled at rate 3/4. The file was then repeated as in a traditional carousel.
- "MPE-FEC 7/8 + Raptor" - MPE-FEC was enabled at rate 7/8 and the file was then sent using AL-FEC (Raptor) coding.
- "No MPE-FEC + Raptor" - only Raptor coding was used.

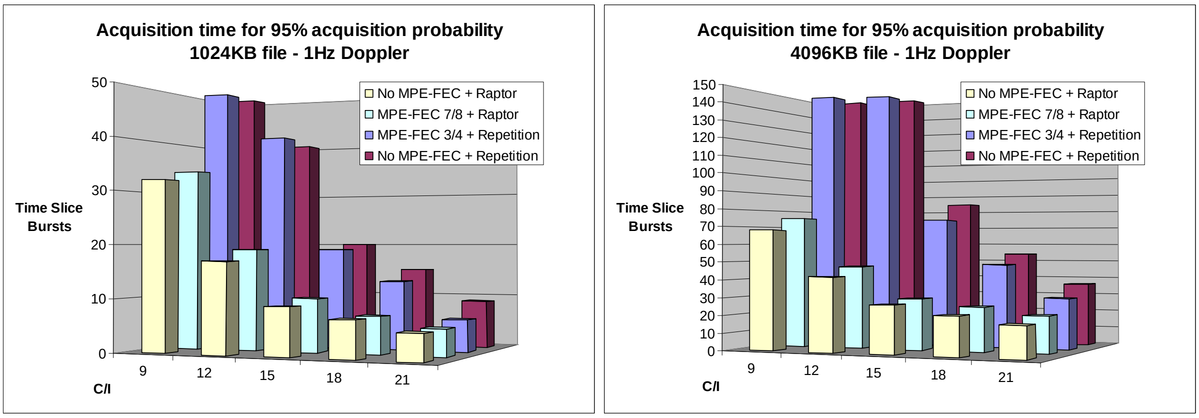

Figure A.1: 1 Hz Doppler

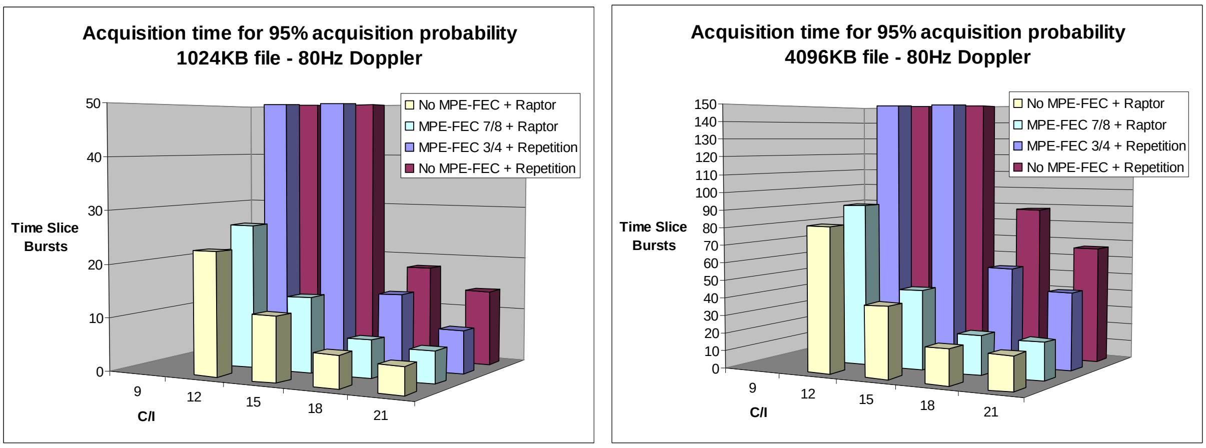

Figure A.2: 80 Hz Doppler

These results are repeated in tabular form in tables A.2 and A.3.

Table A.2: Delivery times (Time Slice Bursts) for 1 MB file

C/N	No MPE-FEC + repetition		MPE-FEC 3/4 + repetition		MPE-FEC 7/8 + Raptor		No MPE-FEC + Raptor	
	1 Hz	80 Hz	1 Hz	80 Hz	1 Hz	80 Hz	1 Hz	80 Hz
9	n/a	n/a	n/a	n/a	34		32	
12	40	n/a	41	n/a	19	27	17	23
15	20	n/a	19	n/a	10	14	9	12
18	15	18	13	14	7	7	7	6
21	9	14	6	8	5	6	5	5

Table A.3: Delivery times (Time Slice Bursts) for 4 MB file

C/N	No MPE-FEC + repetition		MPE-FEC 3/4 + repetition		MPE-FEC 7/8 + Raptor		No MPE-FEC + Raptor	
	1 Hz	80 Hz	1 Hz	80 Hz	1 Hz	80 Hz	1 Hz	80 Hz
9	n/a		n/a	n/a	75		68	
12	171	n/a	149	n/a	47	92	42	82
15	84	n/a	74	n/a	29	45	27	40
18	54	88	48	56	25	22	22	20
21	36	66	29	44	21	21	18	19

Figures A.3 and A.4 show similar results for 128 KB and 512 KB files. In these cases results for "MPE-FEC 3/4 + Repetition" and "No MPE-FEC plus Raptor" only were available.

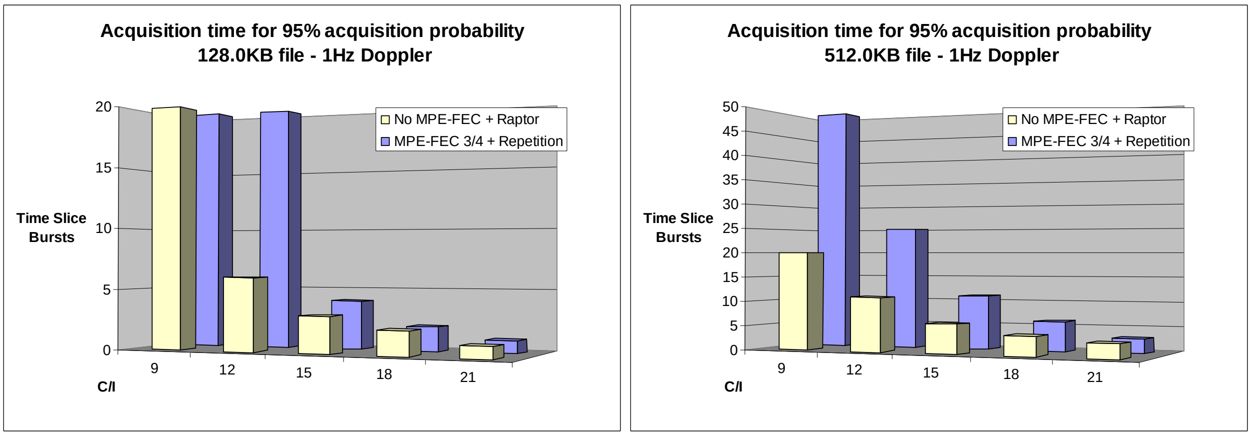

Figure A.3: Small files, 1 Hz Doppler

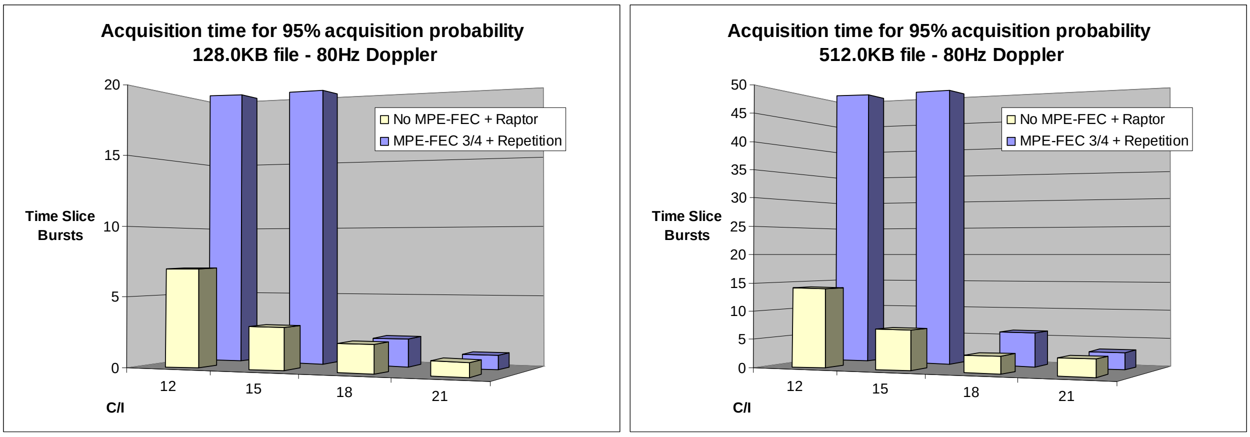

Figure A.4: Small files, 80 Hz Doppler

These results are repeated in tabular form in tables A.4 and A.5.

Table A.4: Delivery times (Time Slice Bursts) for 128 Kb file

C/N	MPE-FEC 3/4 + repetition		No MPE-FEC + Raptor	
	1 Hz	80 Hz	1 Hz	80 Hz
9	n/a		n/a	
12	n/a	n/a	6	7
15	4	n/a	3	3
18	2	2	2	2
21	1	1	1	1

Table A.5: Delivery times (Time Slice Bursts) for 512 Kb file

C/N	MPE-FEC 3/4 + repetition		No MPE-FEC + Raptor	
	1 Hz	80 Hz	1 Hz	80 Hz
9	n/a		20	
12	25	n/a	11	14
15	11	n/a	6	7
18	6	6	4	3
21	3	3	3	3

History

Document history		
V1.1.1	October 2007	Publication
V1.2.1	June 2009	Publication