Enriched Network-aware Video Services over Internet Overlay Networks

www.envision-project.org

Deliverable D5.1

Initial Specification of Metadata Management, Dynamic Content Generation and Adaptation

Public report, Version 2, 19 May 2011

Authors

UCL David Griffin, Eleni Mykoniati, Miguel Rio, Raul Landa

ALUD -

LaBRI Toufik Ahmed, Abbas Bradai, Ubaid Abbasi, Samir Medjiah

FT Bertrand Mathieu, Sylvain Kervadec, Stéphanie Relier, Pierre Paris, Bastide Valery

TID Oriol Ribera Prats

LIVEU Noam Amram

Reviewers Nico Schwan, Klaus Satzke, David Griffin

Abstract This deliverable presents work achieved in workpackage 5 (WP5) during the first year of the ENVISION project. It provides a detailed state of the art and an initial specification of ENVISION metadata management, content generation and content adaptation techniques. Our aim is to study and develop methods to adapt audio-visual (AV) content, *on-the-fly*, to the available network resources, and in accordance with the characteristics and capabilities of users and terminals. To this end, we present a user profile model that reflects both user preferences and the terminal and network characteristics that can influence AV content encoding and its transmission over the overlay network and its underlying ISP infrastructure. In addition, this deliverable presents initial studies and specifications for the use in ENVISION of techniques for the delivery of high–QoS multimedia content, such as forward error correction, caching and mechanisms to boost the average upload throughput.

Keywords Metadata, AV Content Generation, AV Content Adaptation, Error Resilience, Forward Error Correction, Caching, Multilink Delivery.

Project funded by the European Union under the
Information and Communication Technologies FP7 Cooperation Programme
Grant Agreement number 248565

EXECUTIVE SUMMARY

This deliverable describes mechanisms, techniques and algorithms for content generation and adaptation that support the delivery of multimedia content to a large number of end-users. These techniques are designed to consider not only the capabilities of users, but also the changing conditions of the network used for content delivery.

This deliverable will investigate the use of media adaptation to improve the Quality of Experience experienced by the end-user. In particular, the D5.1 provides the following contributions:

- A profile model for users, terminals, content, and offered services, to be used by content generation, consumption, and adaptation processes;
- An initial specification of metadata and profile management processes;
- A study of candidate content generation techniques such as H.264 AVC (Advanced Video Coding) and SVC (Scalable Video Coding), and its possible uses within ENVISION;
- An initial specification for a QoS-based cross-layer adaptation engine that performs both decision and adaptation operations;
- An initial specification for AV transmission techniques that provide error resiliency and enhance the stream reliability. These techniques take into account various parameters that can be provided through the CINA interface, such as available bandwidth or path error rate, along with additional information, such as required resiliency or packet priority;
- An initial specification for a scalable caching service that allows content to be distributed to a large number of viewers without commensurate increases in the quantity of dedicated resources;
- An initial specification for mechanisms to boost the average upload throughput of the participants by relying on multilink delivery of AV content.

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

The high-level objective of ENVISION is to enable future media applications to run more efficiently over the Internet, providing adequate Quality of Experience (QoE)/Quality of Service (QoS) for the end-users while maintaining cost-efficiency for the involved business stakeholders. To achieve this high-level objective, ENVISION will: (1) build a cross-layer architecture enabling the cooperation and optimisation of the application and network level functions, (2) develop mechanisms at the application layer to handle the interactive, high-volume content generated and distributed among users with heterogeneous access means, minimising the associated resource requirements while maintaining high QoE/QoS, and making use of the resources of the network and of the users themselves, (3) develop mechanisms at the network layer to provide strategic information and resources to the overlay application with the purpose of optimising network management, and to mobilise the network resources where needed.

The main goal of this deliverable, within WP5, is to provide initial specifications for content generation and adaptation techniques that consider the capabilities of end user terminals and the networks used for media delivery. This deliverable also provides a comprehensive review of the state of the art in media adaptation techniques; this will be the basis of a taxonomy of candidate techniques that can be used in the context of the ENVISION project. The presentation of this material proceeds as follows. First, we present a metadata model for users, terminals, content, offered services and both network and overlay characteristics. Then, we investigate mechanisms and techniques to manage and process metadata based on this model. We continue with an investigation of various content generation techniques which are candidates for adoption within the ENVISION project (we focus mainly on H.264 Advanced Video Coding and its scalable extension, Scalable Video Coding). In addition, we propose an initial design and specification for a QoS-based cross-layer adaptation engine that allows the maintenance of an acceptable quality level for the end user. Finally, we study techniques that support large-scale content delivery such as error-resilient AV transmission techniques, caching techniques that improve network efficiency, and mechanisms to boost the average upload throughput using multilink delivery.

This deliverable is organised in 7 sections, as follows. Section 2, “Initial Specification of Metadata Management”, aims to model metadata profiles and their necessary management processes. Section 3, “Content Generation”, investigates different content candidate formats which can be adopted in ENVISION in the context of dynamic content generation. Section 4, “Content Adaptation”, deals with the adaptation of the AV stream to changing contexts and network element profiles. After presenting a general problem formulation, we investigate several important practical questions and suggest potential methods to answer them. Section 5, “Error Resilient AV Transmission”, aims to enhance the reliability of the media stream by using error correction techniques. In this section, we will especially investigate Forward Error Correction (FEC) techniques which are candidates to be used in ENVISION. Section 6, “Content Caching”, deals with caching techniques used to store a set of dynamically changing video segments, and to enable their scalable delivery to a large number of viewers. In addition to investigating existing caching techniques, we present a cooperative caching technique which is a candidate for use in ENVISION. Section 7, “Multilink Enabled Peers for Boosting Content Delivery”, deals with mechanisms to boost the average upload throughput of the participants generating or relaying content by exploiting parallel multilink transmissions. Finally, section 8 concludes this deliverable and presents on-going work.

2. INITIAL SPECIFICATION OF METADATA

2.1 Introduction

Metadata is data about the data [MMA02]. It describes the data sufficiently well as to be used as a surrogate for the data when making decisions regarding description and use of the data. Metadata can give complex information concerning structure description, semantics and content of data items, their associated processes and, more widely, the respective domains of this various information. Metadata are:

- Data describing and documenting data,
- Data about datasets and usage aspects of it,
- The content, quality, condition, and other characteristics of data.

The documenting role of metadata is fundamental. This information can give decision elements in order to choose the most appropriate dataset and the most appropriate data presentation mode. In the case of large amounts of data, it is difficult to analyse data content in a straight way. Metadata then gives appreciation or description elements of the information in the dataset.

However, metadata role is not restricted to documenting information. Metadata must also allow:

- Data acquisition and transformation that are complex steps for data producers.
- Metadata can, on one hand, represent the production memory by describing operations carried out during data acquisition and transformation process, and it can, on the other hand, prevent a data producer from repeating the production step of an already existing dataset.
- Description of structure and role of data, in order to allow its interpretation and treatment by a user, especially during transfer steps.

In ENVISION, metadata has an essential role in describing principal aspects of video content management from content generation to consumption. This metadata should capture also the state of all the components involved in the content distribution chain: user information and preferences, usage history, presentation preferences, device and codec capabilities, adaptation capabilities, type and description of services, network description, etc. It allows the creation of context-based multimedia services that maintains an acceptable level of QoE/QoS for the end-consumer. It allows also the service to be accessible anytime, from any location and using any terminal device, and the convergence of different heterogeneous distribution channels.

The key elements of the metadata described and managed by ENVISION includes (but are not limited to):

- End user metadata for describing the end user profile (such as the user, its preferences and its media usage history, etc.)
- Terminal capabilities metadata describing technical properties of the terminal both for source and destination users (such as codec and display capabilities, etc.)
- Content description metadata to characterise the content exchanged between users (such as AV characteristics, spatio-temporal context of the content, Intellectual property etc.)
- Network metadata to describe the parameters of the network (such as network capabilities and conditions, available bandwidth, etc.)
- Metadata for service management (such as service type, service cost, etc.)
- Session description metadata (such as session start time, number of active sessions, etc.)

- Peer metadata to describe the overlay functionalities of a peer (such as adaptation, caching capabilities, etc.)

2.2 State of the Art

The modelling of metadata is an important ingredient of the seamless integration of enriched multimedia content, its generation and its delivery over heterogeneous network infrastructure. The most used existing standards for modelling of metadata are Dublin Core [RFC2413], MPEG-7 [MSS02] MPEG-21 [B03], [CSP01], and TV-Anytime [PS00].

Dublin Core provides a basic description of AV content, while MPEG-7 standardises a number of content description tools that allow effective indexing and retrieval of video content. These tools include video segment description tools, textual annotation and transcription description tools, feature description tools, semantics and model description tools. There are also standards that are specific to certain content categories. For example, for the description of news, the industrial standard NewsML 1.2 [NSM], an XML mark-up language, is widely used for tagging multimedia news items. The MPEG-21 specifications focus on the delivery of media across heterogeneous networks and terminals. There is still a lot of progress to be made in order to enhance metadata to support the new interactive, multi-participant overlay applications in terms of: metadata modelling, procedures for their dynamic update and content adaptation based context and profiles, semantic and perceptive user preferences, lightweight metadata for embedded devices, etc. We first start by providing an overview of the wide used existing metadata standards.

2.2.1 Dublin Core Metadata

Dublin Core (DC) [RFC2413] was designed specifically for generating metadata to facilitate the resource discovery of textual documents. The "pure Dublin Core" approach provides multiple levels of descriptive information. At the top level, the fifteen basic Dublin Core elements can be used to describe the fundamental bibliographic type information about the complete document (Table 1). This enables non-specialist inter-disciplinary searching, independent of the media type.

Element	Definition
Title	A name given to the resource
Creator	An entity primarily responsible for making the content of the resource
Subject	The topic of the content of the resource
Description	An account of the content of the resource Description may include but is not limited to: an abstract, table of contents, reference to a graphical representation of content or a free-text account of the content
Publisher	An entity responsible for making the resource available
Contributor	An entity responsible for making contributions to the content of the resource
Date	A date associated with an event in the life cycle of the resource. Recommended best practice for encoding the date value is defined in a profile of ISO 8601 and follows the YYYY-MM-DD format.
Type	The nature or genre of the content of the resource. Type includes terms describing general categories, functions, genres, or aggregation levels for content.
Format	The physical or digital manifestation of the resource. Typically, Format may include the media-type or dimensions of the resource. Format may be used to determine the software, hardware or other equipment needed to display or

	operate the resource. Examples of dimensions include size and duration. Recommended best practice is to select a value from a controlled vocabulary (for example, the list of Internet Media Types defining computer media formats).
Identifier	An unambiguous reference to the resource within a given context. Example formal identification systems include the Uniform Resource Identifier (URI) (including the Uniform Resource Locator (URL)), the Digital Object Identifier (DOI) and the International Standard Book Number (ISBN).
Source	A Reference to a resource from which the present resource is derived
Language	A language of the intellectual content of the resource. Recommended best practice for the values of the Language element is defined by RFC 1766 [RFC1766][RFC1766] which includes a two-letter Language Code (taken from the ISO 639 standard [ISO639]), followed optionally, by a two-letter Country Code (taken from the ISO 3166 standard [ISO3166]).
Relation	A reference to a related resource
Coverage	The extent or scope of the content of the resource. Coverage will typically include spatial location (a place name or geographic coordinates), temporal period (a period label, date, or date range) or jurisdiction (such as a named administrative entity). Recommended best practice is to select a value from a controlled vocabulary (for example, the Thesaurus of Geographic Names [TGN]) and that, where appropriate, named places or time periods be used in preference to numeric identifiers such as sets of coordinates or date ranges.
Rights	Information about rights held in and over the resource

Table 1: Dublin Core metadata

The reason why DC is so commonly referenced lies in the fact that it provides a baseline metadata set that is generally applicable to all kinds of data. The extensions or qualifiers to specific DC elements (*Type, Description, Relation, Coverage*) can be applied at the lower levels (scenes, shots, frames) to provide fine-grained, discipline-and media-specific searching (e.g. *Description.Camera.Angle*).

As shown in Figure 1, it is possible to describe both the structure and fine-grained details of video content by using the fifteen Dublin Core elements plus qualifiers and encoding this within Resource Description Framework (RDF) [HI98].

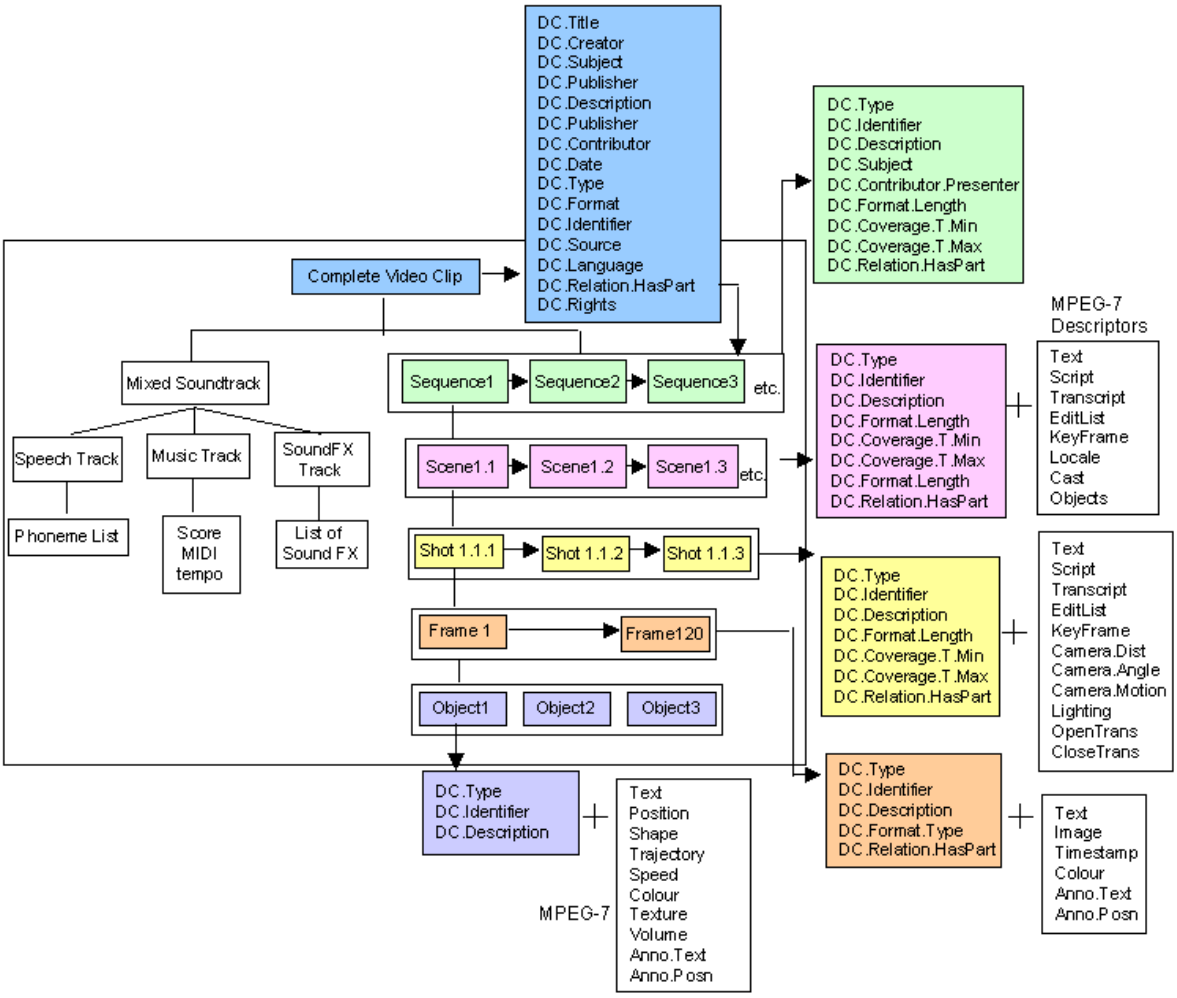


Figure 1: Multi-layered Hierarchical Structure and Attributes of Video

2.2.2 MPEG-7

While MPEG-1, MPEG-2 and MPEG-4 standards focus on coding/decoding and representation of AV content, MPEG-7 [MSS02][CSP01] focuses on description of multimedia content. Its ultimate goal is to provide interoperability among different, heterogeneous systems and applications, management, distribution, and consumption of AV content descriptions. It helps users or applications to search, identify, filter and browse multimedia information.

2.2.2.1 MPEG-7 Structure

The MPEG-7 specification includes a standardised set of descriptors (Ds) and Descriptors Schema (DSs) for audio, visual, multimedia, and a formal language for defining DSs and Ds (Description Definition Language). Formally, the MPEG-7 is organised into the following parts:

2.2.2.1.1 Descriptors (D)

Descriptors define syntax and semantics of features of AV content. Ds may include shape, motion, texture, colour, content genres, events, camera motion, etc.

2.2.2.1.2 Description Schema (DS)

It specifies the structure and semantics of their components, which may be descriptions data type (Descriptors) or even other Description Schemes. Furthermore, the Description Metadata DS describes metadata, such as creation time, extraction instrument, version, confidence, and so forth.

2.2.2.1.3 Description Definition Language (DDL)

The description and definition language defines a set of syntactic, structural, and value constraints to which, a valid MPEG-7 descriptors, description schemes, and descriptions must conform. It permits the creation of new DSs, Ds, along with the modification and extension of existing ones.

2.2.2.1.4 MPEG-7 Systems

MPEG-7 Systems define system level functionalities for the compact, dynamic transmission of MPEG-7 descriptions, such as preparation of MPEG-7 descriptions for efficient transport/storage, synchronisation of content and descriptions, and development of conformant decoders. In addition, MPEG-7 systems describe smart methods for XML binarization (BIM)

2.2.2.2 MPEG-7 Multimedia Description Schemes

In this sub-section, we provide an overview of the MPEG-7 Multimedia Description Schemes (DS) and describe their targeted functionality and use in multimedia applications. DSs are organised as follow:

2.2.2.2.1 Basic Elements

The basic elements provide a set of data types and mathematical structures such as vectors and matrices, which are needed by the DSs for describing AV content. The basic elements include constructs for linking media files, localising pieces of content and describing time, places, persons, individuals, groups, organisations, and other textual annotation.

2.2.2.2.2 Content Description

MPEG-7 provides DSs for description of the structure and semantics of AV content. The structural part describes the video segments, frames, and moving regions and audio segments. The semantic part describes the objects, events, and notions from the real world that are captured by the AV content.

2.2.2.2.3 Content Management

MPEG-7 provides DSs for multimedia content management. These tools describe the following information:

- Creation and production: The *Creation Information* provides a title, textual annotation, and information such as creators, creation locations, and dates. The classification information describes how the AV material is classified into categories such as genre, subject, purpose, language, and so forth. It provides also review and guidance information such as age classification, parental guidance, and subjective review.
- Media coding, storage and file formats: the *Media Information* describes the storage media such as the format, compression, and coding of the AV content.
- Content usage: The *Usage Information* describes the usage information related to the AV content such as usage rights, usage record, and financial information.

2.2.2.2.4 Navigation and Access

MPEG-7 provides also DSs for facilitating browsing and retrieval of AV content by defining summaries (e.g. Key frames), partitions and decompositions (temporal and spatial decomposition) and variations of the AV material.

2.2.2.2.5 User Interaction

The *User Interaction* DS describe preferences of users pertaining to the consumption of the AV content, as well as usage history, in order to select and personalise AV content for more efficient access, presentation and consumption.

The *Usage History* DS describes the history of actions carried out by a user of a multimedia system. The *Usage History* descriptions can be exchanged between consumers, their agents, and content providers, and may in turn be used to determine the user preferences with regard to AV content.

2.2.2.2.6 Multimedia Descriptors

Multimedia descriptors define:

- *Visual Descriptor*: The visual features described by MPEG-7 are colour, texture, shape and motion.
- *Audio Descriptors*: MPEG-7 audio standardises some low-level descriptors that are used both as common building blocks and for advanced tools such as search and retrieval of spoken content.

2.2.3 MPEG-21

The goal of MPEG-21 [B03] is to define the technology needed to support users to exchange, access, consume, trade and manipulate digital items in an efficient, transparent and interoperable way. MPEG-21 is based on two essential concepts: the definition of a fundamental unit of distribution and transaction (the Digital Item) and the concept of users interacting with Digital Items. Additionally, MPEG-21 defines a "Right expression Language" standard as a mean of sharing digital rights/permissions/restrictions for digital content from content creator to content consumer. Digital Item Adaptation (DIA) metadata is another key element of MPEG-21 standard, used to facilitate the adaptation of the DI.

2.2.3.1 Structure of MPEG-21

2.2.3.1.1 Digital Item

A DI (Figure 2) is a structured object with a standard representation; it is a combination of resources, metadata, and structure. The metadata comprises informational data about the Digital Item as a whole or to the individual resources included in the Digital Item. Finally, the structure relates to the relationships among the parts of the Digital Item, both resources and metadata.

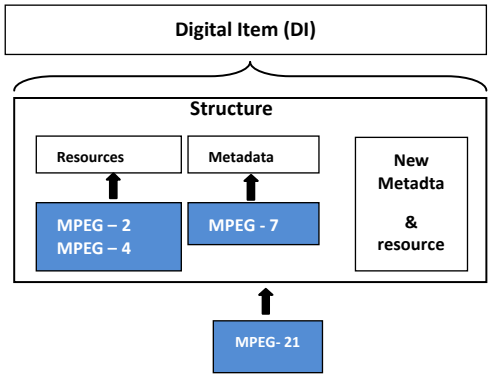


Figure 2: Digital Item Structure

2.2.3.1.2 Digital Item Declaration (DID)

The purpose of the Digital Item Declaration (DID) specification is to describe a set of abstract terms and concepts to form a useful model for defining Digital Items.

We list bellow some important terms of the model:

- *Item*: is a group of sub-items and/or component. Items are declarative representations of Digital Items.

- *Descriptor*: a descriptor associates information with the enclosing element. This information may be a component (such as a thumbnail of an image, or a text component), or a textual statements.
- *Container*: is a structure that allows items and/or containers to be grouped.
- *Resource*: a resource is an individually identifiable asset such as a video or audio clip, an image, or a textual asset. A resource may also potentially be a physical object. All resources must be locatable via an unambiguous address.
- *Component*: information related to all or parts of the specific resource instance, it contains control or structural information about the resource.

2.2.3.1.3 Digital Item Identification (DII)

The scope of the Digital Item identification is how to:

- Uniquely identify Digital items and parts of them.
- Uniquely identify Intellectual property (IP) related to the Digital Items (and part of them).
- Use identifiers to link Digital Items with related information such as descriptive metadata.
- Identify different types of Digital Items.

For content identification, the DII and DID provides the ability to associate Uniform Resource Identifiers (URIs) with an entire Digital Item or its parts. For content description, the DII and DID framework provides the ability to include metadata from various sources and in various formats including XML or plain text.

2.2.3.1.4 MPEG Intellectual Property Management and Protection Extension Metadata (IPMP)

MPEG-21 handles the intellectual property by extending the Intellectual Property Management and Protection (IPMP) defined in MPEG-2 and MPEG-4. MPEG-21 standardises ways to retrieve IPMP tools from remote location, exchanging messages between IPMP tools and between these tools and the terminal. It also dresses authentication of IPMP tools.

2.2.3.1.5 Right Expression Language (REL)

A Right Expression Language is designed for the licensing of digital materials, especially video and audio. The REL is intended to provide flexible, interoperable mechanisms to support transparent use of digital resources in publishing, distributing and consuming of digital movies, digital music, electronic books, broadcasting, interactive games, and computer software, in a way that protects digital content and honours the rights, conditions, and fees specified for digital content.

The basic construct of a Right Expression Language is the rights expression, which describes a permission granted to a user or consumer of protected content. The expression can be very simple, such as, “this content may be printed/played” or very complex, such as, “this content can be played on Monday 15th February at 2.00 p.m.”, provided that the device meets the following criteria..., e.g. “the device must be equipped with a secure processor” and the user has contacted www.mywebsite.com first in order to provide the details, e.g. name, address, age.

2.2.3.1.6 Digital Item Adaptation (DIA)

The Digital Item adaptation (DIA), referred in the 7th part of MPEG-21 standard (ISO/IEC 21000-7), aims to achieve interoperable and transparent access to multimedia content, hiding the specifics of terminal devices, networks and content formats. In the next section we discuss in detail the organisation and functions of the DIA.

2.2.3.2 Focus on the Digital Item Adaptation

Figure 3 shows the general DIA concept: A Digital Item is subject to both a resource adaptation and a descriptor adaptation engine, which together produce the adapted Digital Item. Note that the standard specifies only the tools that assist with the adaptation process, not the adaptation engines themselves. DIA tools are organised in five major categories:

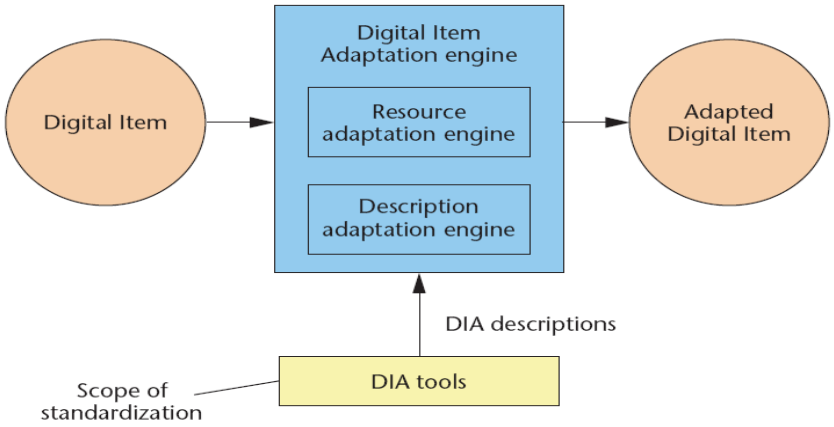


Figure 3: Digital Item Adaptation Architecture

2.2.3.2.1 Usage Environment Description Tools

The usage environment description tools describe the terminal capabilities as well as characteristics of the network, user, and natural environment such as lighting conditions or noise level, or a circumstance such as the time and location. These tools allow the generation and manipulation of descriptive information concerning the usage environment, which are used to adapt DIs for transmission, storage and consumption. We describe below, some potential uses for these descriptions.

2.2.3.2.1.1 Terminal Capabilities

In addition to enabling media format compatibility, the terminal capabilities description allows users to adapt various forms of multimedia for consumption on a particular terminal. DIA specifies the following class of descriptors:

- *Codec capabilities*: specify the format that a particular terminal is capable of encoding or decoding, ex: MPEG-2, MPEG-4, etc.
- *Input-Output capabilities*: include a description of display, audio capabilities, and various properties of several input device types.
- *Device properties*: describe power attributes of the device, as well as I/O characteristics.

2.2.3.2.1.2 Network Characteristics

These specifications consider two main categories: network capabilities and network conditions. Network capabilities define a network's static attributes, such as the maximum capacity of a network and the minimum guaranteed bandwidth, while network conditions describe network parameters that tend to be more dynamic such as the available bandwidth, error and delay characteristics.

2.2.3.2.1.3 User Characteristics

User characteristics define the following elements:

- User general characteristics, user preferences and usage history, which have been imported from MPEG-7.
- Accessibility characteristics provide descriptions that enable users to adapt content according to certain auditory and or visual impairment.
- Presentation preferences specify preferences related to the means by which audiovisual information are presented to the user, as preferred audio power, equaliser settings, the preferred colour degree, brightness, and contrast.
- Location characteristics include mobility and destination descriptions.

2.2.3.2.1.4 Natural Environment Characteristics

The Digital Item Adaptation describes the natural environment characteristic. It specifies:

- Digital Item's location and time of usage, imported from MPEG-7 description scheme (DS)
- Audiovisual environment, such as the noise level and the illumination characteristics that may affect the perceived display of visual information

2.2.3.2.2 Terminal and Network QoS Tools

The set of terminal and network quality of service tools defined by DIA specify information that would help users to decide the optimal adaptation strategy, it allows the generation of metadata describing the relationships between QoS constraints, adaptation operation and the expected results.

2.2.3.2.3 BitStream Syntax Description Tools

The bitstream syntax description tools define the syntax of the coding format of a binary media resource. This description allows an adaptation engine to transform the bitstream without having to understand the subtleties of any specific media format.

2.2.3.2.4 Metadata Adaptability Tools

The metadata adaptability tools specify information that can be used to reduce the complexity of adapting metadata.

2.2.3.2.5 Session Mobility

In DIA, session mobility is the transfer of configuration-state (information concerning the consumption of a DI) in a device onto another device.

2.2.4 TV-AnyTime

TV-anytime [PS00] is a set of specifications for the controlled delivery of multimedia content to a user's personal device (Personal Video Recorder (PVR)), it seeks to exploit the evolution in storage capabilities in consumer platforms and help the development of interoperable, integrated and secure systems from content providers, through service providers to the consumers. Users will have access to content from a wide variety of resources (broadcast, online resources...), adapted to their needs and personal preferences.

TV anytime provides normative specifications in three areas:

- Content referencing and location resolution for finding content
- Metadata for describing content
- Right management and protection for protecting the content

2.2.4.1 Content Referencing

The purpose of content referencing is to provide a consistent way to reference content from different sources (terrestrial TV, satellite, cable, Internet, etc.) independently from its location. The separation is provided by Content Reference IDentifier (CRID), which references a content despite of it location, delivery network (cable, Internet, etc.) and independent of the delivery protocol used. Additionally, CRID allows defining a reference to a group of programs as series, and retrieving a specific instance of a specific item of content (unlike ISBN which references content but not a specific copy of the content).

2.2.4.2 TV-Anytime Metadata

The principal idea of TV-AnyTime metadata is to describe content such that user, or a device, can understand what content is available and able to acquire it. The specification is defined in terms of an XML Schema.

The specification defines a document structure which contains a several parts describing information about: programs, group of programs, how programs may be segmented, the preference of the user, etc. We can group this information into four basic kinds of metadata:

- Content description metadata
- Instance description metadata
- Consumer description metadata
- Segmentation metadata

2.2.4.2.1 Content Description Metadata

Content description metadata is divided into four areas:

ProgramInformationTable: Descriptions of the items of content e.g. television program, they include things like the title of the programme, synopsis, genre, and a list of keywords that can be used to match a search.

GroupInformationTable: Descriptions of group of related items of content, such as all episodes of a series.

CreditInformationTable: provides a unique identifier for members, it allows to avoid a multiple spelling of the same person's name. This identifier can be used in other metadata instance.

ProgramReviewTable: provides a critical review of items of content.

2.2.4.2.2 Instance Description Metadata

Instance description metadata is divided into two tables:

- *ProgramLocationTable*: Description of a particular instance of the programme. This metadata contains the scheduled time, duration and service (TV channel)
- *Service Information Table*: Description of service within the system

2.2.4.2.3 Consumer Metadata

Consumer metadata is divided into two areas:
UserPreferences: Details of user's preferences or profile. It provides rich representation of the particular type of content preferred or requested by the user, thus enables user to efficiently search, filter, select and consume desired content.
UsageHistory: Provides a list of the actions carried out by user over an observation period. This information can be used by automatic analysis to generate user preferences.

2.2.4.2.4 Segmentation Metadata

Segmentation refers to the ability to define access and manipulate temporal intervals (segments) within an AV stream. By associating metadata with segments, it is possible to describe the highlights of a program, or the division of programme in chapters as in DVD menu, or put a set of bookmarks in the stream. Such metadata can be provided by service providers or broadcasters as a value-added feature, and/or generated by viewers themselves.

2.2.4.3 Delivery of TV-Anytime Metadata

The delivery of TV-Anytime metadata unfolds in several steps: fragmentation, encoding, encapsulation, and indexing step. The mechanism is summarised in Figure 4.

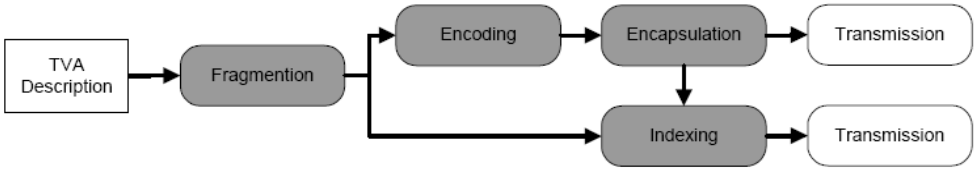


Figure 4: Processing of a TVA Metadata Description for its Delivery over a Unidirectional Link

2.2.4.3.1 Fragmentation

Fragmentation is the decomposition mechanism of a TV-Anytime metadata description into self-consistent units of data, called *TVA fragment*.

A fragment is the ultimate atomic part of a metadata description that can be transmitted independently to a terminal. The fragment must capable of being updated independently from other fragments.

2.2.4.3.2 Encoding

For bandwidth efficiency the TV-anytime metadata fragments must be encoded for transport and delivery. Binary MPEG format for XML (BIM), in association with Zlib compression for textual data, was chosen as the encoding format of TV-Anytime, for the purpose of interoperability.

2.2.4.3.3 Encapsulation

Encapsulation is the process that enables the grouping of encoded fragments in “containers” ready for transmission (Figure 4). It associates furthers information to these fragments such as a unique identifier and a version number.

2.2.4.3.4 Indexing

Indexing is an optional mechanism to deliver TVA metadata to receivers with limited processing and storage capabilities. Indexing information accompanying a fragment stream provides direct access to each fragment.

2.2.5 Conclusion

In this section we have presented the state of the art on metadata standards. We have focused on the standards which are widely used, namely: Dublin Core, MPEG-7, MPEG-21 and TV-Anytime. Dublin Core based on 15 base fields allows a basic description of AV content. MPEG-7 provides more detailed description of the AV content. More than it provides description of the user preferences on the content and offers tools to manage it. MPEG-21 goes farther than and provides in addition to MPEG-7 descriptors and tools, description of the usage environment and proposes tools to perform content adaptation (DIA). TV-anytime focuses on the content and allows description, referencing and location resolution for finding the content. In addition, it proposes content right management and protecting tools. Nevertheless, many of metadata elements described in this standards are out of

scope of ENVISION and many others need to be extended to meet ENVISION requirements. In addition, neither the description of the P2P functionalities of the network nor the cooperative aspect between the application and the Network was addressed in these metadata standards. For these reasons, and in order to optimise processing, transport and update of ENVISION metadata, we propose to design an ENVISION metadata model based on some elements from MPEG-21 and MPEG-7 standards that satisfy the ENVISION needs and use cases.

2.3 ENVISION Metadata Requirements

2.3.1 Metadata Structure Requirements

The selection and the integrated management of metadata depend on the use cases proposed by ENVISION, previously defined in D 2.1 [D2.1].

In ENVISION we have defined mainly three scenarios: Web 3D conference, Bicycle race and Legacy Delivery Networks.

The first scenario deals with the Web 3D conference. Virtual meetings are slated to replace more and more physical meetings. Indeed the virtual meeting and event technologies allow to reach the two main objectives of real live meeting, namely to exchange information and networking, with important saving by wiping out the costs of venue rental, accommodation, transportation, and by reducing the ecological footprint of such a travelling. Because of the dynamicity of the use-case (change of speaker, audience, small groups, exchange of information, mobility of users, etc.), the ENVISION approach is challenged in order to deliver content with good quality of service to the participants.

The second scenario is “Bicycle race”. A live sports event often spans large geographical regions, including both urban and rural areas. In this scenario the competitors, their management teams, and camera teams are constantly moving, while the spectators are spread along the race track and have visual contact with the competitors only once or a few times during the race. This means that the bandwidth requirement for the affected network segments will constantly change as the race progresses. Due to the large geographic distribution, the available networks might be provided by different operators. In such difficult conditions ENVISION aims to provide a good network services and applications operating.

The third scenario is “Legacy Delivery Networks”. In this scenario, the Delivery Network (DN) consists of an overlay service where content, service or application providers decide to provision stored or live content to be later consumed by users, in a way such that the network resources employed are optimal. The main goal of this service is to significantly reduce the network costs of multimedia services while delivering services to a large number of users. Content in a DN follows a life cycle composed of the following stages:

- Reservation stage: The content provider reserves the appropriate resources and provides descriptive information about the content and the providers’ channels.
- Ingestion stage: The provider uploads the content or provides a descriptor of the live content.
- Publishing stage: The provider inserts content and its providers’ information in the suitable website, news feeds, Internet platforms, etc.
- Consumption stage: The consumer visits the web page where the content links are embedded. The browser then fetches the content from the DN. The DN, based on the content, user and environment metadata, decides the most appropriate source from which the browser can obtain the content.

Table 2, Table 3 and Table 4 synthesise ENVISION use cases scenarios and the corresponding metadata elements. For each use case scenario we describe its main features and the corresponding required metadata elements.

Use case feature description	Required Metadata Elements
Multiple content streams are delivered to several end user devices (TV, computer, mobile) simultaneously	• Terminal description: device class, network interface(s), user interaction input, capture interface characteristics, codec capability, codec parameters, display capabilities, Audio/Video output capabilities, power characteristics, memory types and its performances, CPU performances • Content description: content name (identifier), content type (audio, video content), textual description (short description + keywords), input audio/video formats and resolution, output audio/video formats and resolution, output Audio/video bitrates. • Session description: session identifier, number of current active sessions...
Users roaming from one terminal to another possibly changing providers	• Terminal description • Network description: maximum bit Rate, delay, bitErrorRate, packet loss, jitter, QoS mechanism supported … • Service description: Service ID, service cost, service provider storage capacity, service provider available storage pace, service codec preferences, list of Content identifier, service description … • End user description: general information, virtual information, authentication information, localisation information, audio/display presentation preference, usage history, user class: simple, premium, etc. Adaptation preference: audio first, video first, spatial, temporal, SNR
Video content sources are roaming across many network domains and providers	• Content description • Networks description
Avatar personalisation, even cloning the user through media (multi viewpoint photos)	• User photo, or 3D avatar
Attend an event in a shared room, watch the live video of the presenter	• Conference start time, duration
Allow the presenter to control the slide show, white board, and play recorded video as part of the presentation broadcasted to everybody	• User rights on content
On demand access to recorded content related to the event (relevance is defined at	• Content description

Use case feature description	Required Metadata Elements
the application layer)	
Ad-hoc creation of conversational conference sessions (video, audio) between multiple participants	• Session description • End user description
User registration enabling the provider to pre-provision resources	• Terminal description • User history preferences
Content adaptation (server-side, proxy, or user-side) suitable for heterogeneous access capabilities (3D virtual world and/or video coding) based on static profiles and dynamic network conditions	• Network description • Terminal description
Content distribution using participant resources in a P2P fashion for background content, participant live video streams, state updates of virtual objects and players, etc.	• Peer functionalities description

Table 2: Web 3D Conferencing Use Case Scenario vs. Metadata Elements

Use case feature description	Required metadata elements
Video content sources are roaming across many network domains and providers	• Content description • Networks/Domains description
Enable prioritisation of participants, e.g. paying customers receive better QoS	• User class: simple, premium users, etc. • QoS support: does ISP manage QoS traffic
Allows the viewers to specify their restrictions/preferences on the format (e.g. HD) of the received video	• User preferences
Interpolation of relocated, viewpoint tagged video sources for 3D navigation of space	• Localisation description (GPS parameters) and localisation history • Timestamp • Angle view
Shared editing of mash up streams by many editors real-time, to produce a number of alternative versions, to be consumed by a number of end users	• End user description • Transcoding capabilities
Enable the end user to send a profile of his terminal to the application so that the application can adjust the characteristics of media to be delivered to this user	• Terminal description
Build the content distribution topology to cope with high churn of input media (video sources come and go) and of users' interest (selection of viewpoints in HD 3D video, navigation	• Viewpoint (angle)

Use case feature description	Required metadata elements
across a virtual world)	
Offering capabilities to support intellectual property preserving techniques	• Permissions (who can use the content: free for private use, for example), Conditions (price per view, for example.), etc.
Content transcoding in nodes other than the content sources	• Transponders parameters: input formats, output formats • Buffers size, packet size, transcoder average speed
ISP providing storage, computing/processing and other service infrastructure resources to the application	• ISP identifier • ISP storage capacities

Table 3: Bicycle Race Use Case Scenario vs. Metadata Elements

Use case feature description	Required metadata elements
Resource reservation	• Content description • Expected QoE • Service provider description
Content ingestion	• Service provider storage capacity • Live source descriptor
Content publishing	• Content description
Content consumption	• Session description • End user description • Network condition description • Terminal capabilities description • Resource occupation information • Content ranking (done by users)

Table 4: Legacy Delivery Networks

2.3.2 Metadata Management Requirements

2.3.2.1 Management of Metadata Lifecycle

Metadata is dynamic in nature, and its lifetime may vary greatly. The dynamics of metadata is poorly supported by existing technologies. Metadata should be maintained up-to-date in a cost-effective manner by adopting efficient fragmentation politics and updating related entities by means of notification mechanisms. In our model, this is addressed by assuming that metadata is stateful and once changes detected they will be propagated using a notification mechanism.

2.3.2.2 ENVISION Metadata Storage and Access Requirements

The ENVISION metadata storage depends on the nature of the metadata. Several metadata element types, such as the user description metadata, will be stored in local databases or file for privacy and security issues. Some metadata such as the content metadata will be conveyed with the data, others such as the network metadata, should be distributed over the network since it is often requested.

In order to keep integrity and validity of this metadata, read/write rights policies should be defined. There are two main issues regarding this point: Firstly, permission to access and manipulate metadata elements should be defined at different levels of granularity (stream, fragment, tag, etc.) depending on the fragmentation policy. Secondly, access control mechanisms should be uniform regardless of differences in the access mechanisms (push/pull, distributed/centralised metadata, etc.)

Table 5 summarises initial recommendations for ENVISION metadata storage and access mode.

Metadata type	Storage type	Access type
User description metadata	Local file/database or in local DBMSs of service providers	Restricted to authorised entities
Terminal capabilities description metadata	Stored in local file system	
Content metadata	Volatile (extracted from the bitstream) for live stream & selectively in DBMS (allowing search & identification) for non-live stream	Restricted
Network metadata	Distributed over the network	Public
Services description metadata	Stored in local DBMSs of service providers	Restricted
Peer overlay functionalities description metadata	Distributed over the network	Public

Table 5: Metadata Storage and Access Requirements

2.4 ENVISION Metadata Structure

After determining the requirements for the metadata specification scheme in the previous section, a set of criteria was derived from these requirements. The initial design of the metadata specification scheme is proposed based on these requirements. The definition of different profiles metadata depends on the use case scenarios explored in the D2.1 and summarised in Table 2, Table 3 and Table 4.

Conceptually, we defined a set of key metadata elements which include (but are not limited to):

- End user metadata for describing the end user profile, preferences and media usage history
- Terminal capabilities metadata describes technical properties of the terminal both for source or destination users
- Content description metadata to characterise the content generated or exchanged between users, its characteristics in terms of structural and semantic information

- Network metadata to describe the parameters of the network that can be used by the overlay content distribution
- Service metadata for service information and management
- Session description metadata
- Peer metadata

Figure 5 resumes the relationship and interactions between different actors in ENVISION architecture and the profiles metadata. An actor may have two actions on the metadata: create/manage metadata or just consume it. The content provider creates and manages the content description, terminal capabilities and peer metadata and it consumes the session metadata. The network provider creates and updates network metadata. However, an ENVISION actor (Overlay network management, content caching, content adaptation and content relaying entities) consumes and updates all the metadata classes.

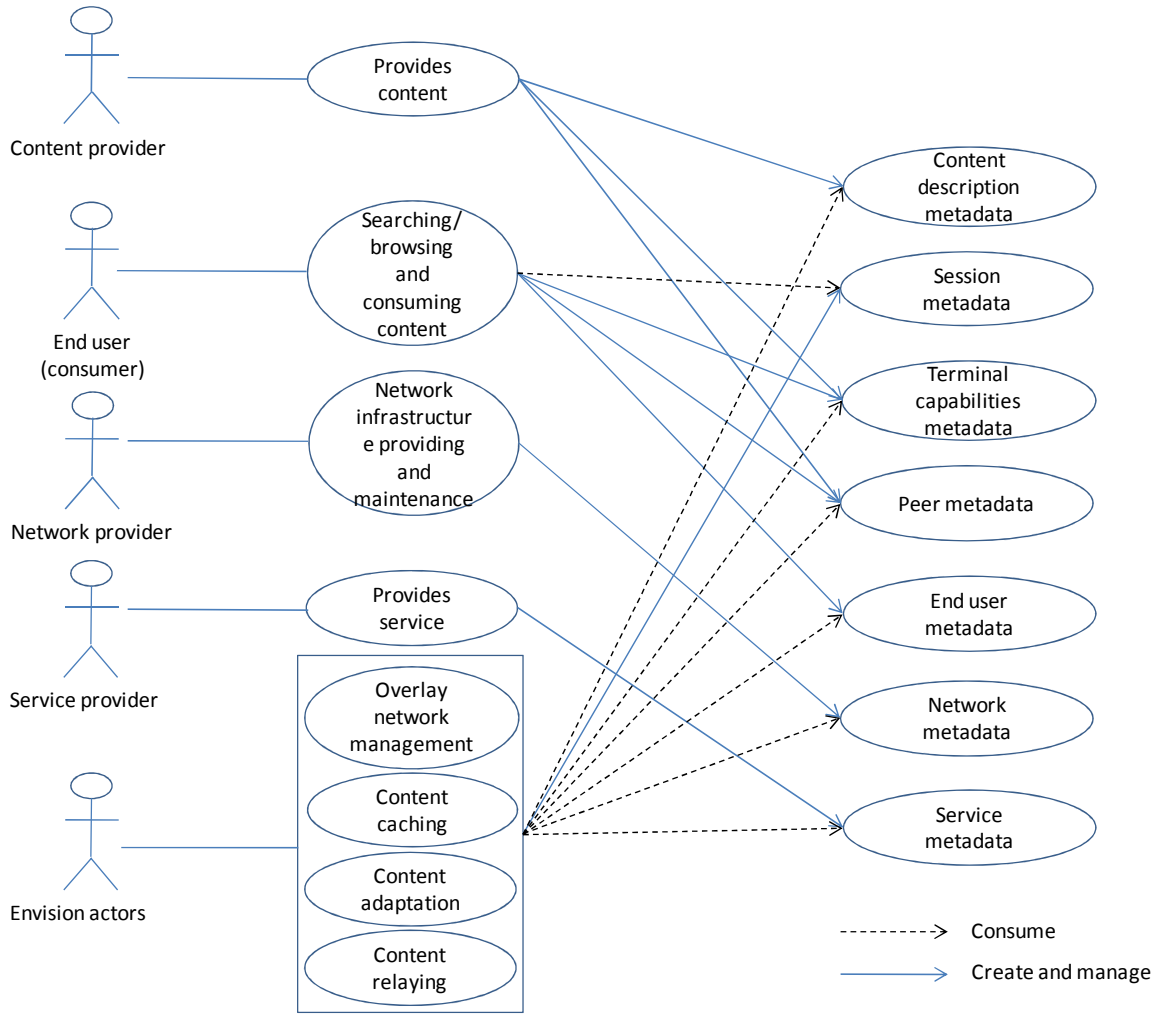


Figure 5: ENVISION Metadata Overview

2.4.1 End User Metadata

The ENVISION terminal shall be able to provide information describing user characteristics and preferences that may be needed in order to provide the required service. A set of descriptors will be used to format such information.

At the moment of editing this document it is not possible to exactly define a closed set of required descriptors for end user. Table 6 summarises and briefly describes those descriptors that at this moment in time seem to be mandatory.

This information will be then be formatted and stored on the local terminal file system, or distributed over the network. It will be accessible by means of queries using an appropriate interface to be defined among the involved entities. For instance, metadata carrying user characteristic can be required by the content provider in order to setup customised offers according to the user's habits. For this purpose, an adequate interface for requesting such information has to be defined between the terminal and the entity requesting the metadata.

Metadata entity	Detail	Description
General information	Name, contact information, photo, status (person, organisation, etc.)	Describes the general characteristics of a user as name and contact information, eventually a photo. A user can be person, group of person or organisation.
Virtual information	Username, avatar, virtual localisation	Describes the general virtual information of a user which is mandatory in contexts where we have to protect the real user information, or it isn't important to disclose it (games context for example).
User class	Simple, premium, etc.	To define user privileges in term of priority, allocated bandwidth, etc.
Authentication information	Public key	For security purposes.
Localisation information	IP address, GPS, Localisation history	A user can be localised by it GPS coordinates (when available), alternatively we can use IP address as an indicator of the user geographical localisation
Display presentation preferences	Colour temperature, brightness intensity, Display orientation, up scaling or not	Describes the preference of a user regarding the colour, brightness of the displayed content.
Audio presentation preference	Volume control, frequency equaliser, preferred language	Describes the preference of a user regarding the audio volume and frequency equaliser
Usage history	Topics of conferences, keywords, connections dates	List history actions of the user: connections dates, subscribed conferences, user area of interest keywords: in order to propose appropriate content to the user (a targeted advertisement of the content)
Adaptation preference	Audio first, video first, spatial, temporal, SNR	The adaptation process must take in consideration the user preferences, some users prefer best video (image) quality rather than audio, others prefer the opposite. Some users prefer lowering the video SNR rather than framerate. Others prefer reduction in spatial resolution rather than reductions in SNR.

Metadata entity	Detail	Description
User rights on content	Read only, modify, record	Define the user's right on the content, it allow answering the question: can user create, delete or modify content?
Other preferences	Preferred network interfaces	

Table 6: End User Metadata Description

2.4.2 Terminal Capabilities Metadata

The ENVISION terminal shall be able to provide information concerning the device platform characteristics and the supported decoding, computing performance and user interaction capabilities. Terminal capabilities include hardware properties such as processor speed, software properties such as operating system, display properties such as screen resolution, and device profiles indicating the supported media formats (e.g. MPEG-2). Based on these metadata, ENVISION adaptation engine adapts the stream to meet the user's terminal characteristics. Table 7 summarises and briefly describes these descriptors that currently seem to be mandatory. The descriptors listed here will be formatted and stored on the local terminal file system.

Metadata entity	Detail	Description
Device class	PC, PDA, laptop, mobile phone, etc.	Specifies the class of the terminal, is it a PC? PDA? laptop?
Network interface(s)	Wired or wireless? Protocol, bandwidth, frequency, coverage area, power consumption	Describes the network interface of the terminal.
User interaction input	Mouse, keyboard, pen, tablet, microphone, etc.	Specifies the various types of user interaction input support that is available on a particular device. With such information an adaptation engine could modify the means by which a user would interact with resources contained in a multimedia presentation.
Capture interface	• Video: output format (codec: MPEG-2, MPEG-4, AVC, SVC, VC-1, etc.), resolution, frame rate. • Audio: output form (codec: AMR, AAC, WMA, etc.) mono, stereo, frequency, bitrates, etc.	It is essential to identify the capture interface capacities of the media source, to select the appropriate stream sources in concordance with the end user specifications and requirements in term of resolution, AV format, etc.
Codec capability	List of supported formats	Describes decoding and encoding capabilities of a terminal. In particular, this metadata element specifies specific formats that a terminal may be capable of

Metadata entity	Detail	Description
		decoding and encoding.
Codec parameters	Buffers size, bitrate	
Supported file format	MP4, MPEG-2 TS, QuickTime, WM, SVC, etc.	Specifies the file formats supported by the terminal.
Delivery protocols	HTTP streaming, RTSP streaming, progressive download, etc.	
Display capabilities	Resolution, maximum brightness, colour depth	Specifies display capabilities of the terminal. Specifically, this metadata provides descriptions of display capabilities elements.
Audio output characteristics	Formats (mono, stereo, etc.), power, signal noise ratio	Specifies audio output capabilities description of the terminal.
Power consumption	Power source, battery capacity	
Processing performance	Processing performance Indicator for the terminal (e.g. High, medium or low processing performance)	Terminals are classified into three classes regarding their processing performance. This can be useful to decide the most suitable node to perform the adaptation for instance.[1]
Transcoding capabilities	Input formats, output formats, transcoding average speed	Specifies the formats that the terminals can transcode and to which formats. This may be useful to implement adaptation facilities.
Storage	Storage capacity, remaining space	Describes the storage capacities of the terminal.
Buffers	Types of buffers, size	

Table 7: Terminal Capabilities Metadata Description

2.4.3 Content Metadata

We describe in Table 8 the content metadata which is mandatory to achieve any content adaptation (transcoding, transrating, etc.), since it allows to describe the AV characteristics of the content such as it audio/video codec and it bitrate. Content metadata provide also a semantic description of the content such as the start time, textual description of the content, subtitle information, etc. Intellectual property protection is supported also in content metadata class via the “intellectual property” field.

More details on this class of metadata are given in Table 8.

[1] The exact performance benchmarks used to classify a terminal depend on the particular application and are beyond the scope of this document.

Metadata entity	Detail	Description
Content identifier		Identifiers are allocated to content in order to uniquely identify it.
Source URL/Filenames		
Start/end time		Specify the start and eventually the end time of the content
Content type	Audiovisual content (A+V+additional media), Audio only, video only, text, html, etc.	Define the type of the content.
Content textual description		Semantic description of the content.
Audio/video/system characteristics	• Audio codec, sample frequency, channel configuration, bitrate, 3D parameters, available translations. • Video codec, resolution, bitrate, colour depth, framerate • Media container, global bitrate, media encryption, etc.	
Content cost	Free content or paid (price)	Determine the cost of the content if it's paid.
Spatio-temporal context of the content		Answer the question where and when it happens? It may be useful to guide the user in his navigation through the different available streams (micro journalism scenarios). Allow to be sure to have a reference time, to facilitate synchronisation.
View angle of the video		Allow users to dynamically navigate through the different angles and viewpoints of the many available streams (micro journalism use case scenario).
Subtitle information	Available subtitle, language information	
Intellectual property	Permissions, conditions	Permissions on the content and Digital Right Management (DRM) Conditions: e.g. you can access the content in return of X€

Table 8: Content Metadata Description

2.4.4 Network metadata

The network characteristics metadata are the subject of the Tasks T3.1, T3.2 and T4.1. In T3.1, the goal is to define which network information (metadata) might be exchanged between the ISPs and the overlay applications while in T3.2 the objective is to define solutions or tools that will help to get the values of such metrics. Typically, the SNMP protocol might be used between the CINA server, hosted by the ISP acting as an SNMP manager and the network equipment of the ISP (e.g. routers, gateways, etc.) acting as SNMP agents providing the values of the monitored metrics. Other solutions for measuring specific metrics are also envisioned. In any way, the network metrics in this context are metrics measured by one ISP within its domain and could be between ingress and egress routers for instance. ISP is able to know for example what is the delay or the bandwidth between two edge routers of its domain. However, it is up to the ISP to provide the information it wishes: some ISPs can provide fine-grained values for many metrics, while others might only reveal coarse-grained information. Network metadata are still under investigation in WP3 to evaluate the ones that are most likely to be exposed by the ISP and useful to the application.

In T4.1 the values of the network metadata metrics correspond to the network performance experienced over an end-to-end path between two overlay nodes. In this case, the metadata may represent measured values between the two endpoints acquired through monitoring at the overlay layer, or estimates that are produced based on network metadata provided by individual ISPs through the CINA interface and the network metadata gathered at the overlay layer.

Table 9 introduces examples of some network metadata that might be useful. The main goal of ENVISION project is to provide the best service quality to users while optimising the network resources consumption. Hence network metadata such as delay and available bandwidth should be taken in consideration in the adaptation process, mostly in applications requiring a good level of QoS, such as web 3D conferencing, where the interactivity of the application is an important element to be considered.

Metadata entity	**detail**	**Description**
Network identifier	ISP/AS ID (autonomous Domain) number	Identifier of the ISP/AS, networks managed by network operators.
QoS mechanisms supported	DiffServ (class of services description), Other (use of RTCP if RTP streaming, which mechanism with HTTP, etc.)?	Allow the adaptation engine to know what adaptation operations are feasible for satisfying the given constraints and the quality requested by a user.
Loss Packet Ratio		Ratio of lost packets on connections.
Nb of hops		Number of hops between access routers.
Available Bandwidth		Available bandwidth between routers according to current traffic.
Maximum Bandwidth		Maximum bandwidth between routers.
One-way delay		Delay between 2 points in one way.

Round-trip delay		Round-trip delay between 2 points.
Error correction		Does the network allow error correction (FEC, etc.)

Table 9: Network Metadata Description

2.4.5 Service Metadata

The service metadata lists the services offered by ENVISION. A service can be live streaming, Video on demand (VoD), video conference, etc. The "*Service ID"* is a unique identifier of a service within a transport stream. Parameters associated with each service referred to are described in this class of metadata such as the service cost, the advised AV parameters for the service, description of the service provider capacities, etc. The different elements of this class of metadata are detailed in Table 10.

Metadata entity	Detail	Description
Service ID		A unique identifier of a service. A service may be a live streaming, Video on Demand (VoD), web conference, etc.
Service cost		We distinguish between the service cost and the content cost, for example we can suggest a VoD service subscription, and a price for each piece of content (a movie, for example).
Service provider storage capacity		
Service provider available storage pace		
Codec preferences		Recommended profiles to best fit the service.
List of Content ID		To be dynamically updated.
Service description		Program start, end, etc.

Table 10: Service Metadata Description

2.4.6 Session Metadata

A session can be defined as a semi-permanent interactive information interchange, also known as a dialogue, a conversation or a meeting, between two or more communicating devices, or between a computer device and user (login). A session is set up or established at a certain point in time, and torn down at a later point in time. An established communication session may involve more than one message in each direction. A session is typically, stateful, meaning that at least one of the communicating parts needs to save information about the session history in order to be able to communicate. In Table 11 the session metadata class elements are described in detail.

Metadata entity	Detail	Description
Session identifier		A unique identifier of the session.
Session start time		
Number of current active sessions		
Max number of active sessions		The maximum number of session that a tracker can manage.
Content identifier		The identifier of the media content managed by the session.
Source peer identifier		
Destination peer identifier		List of destination peer identifiers
Event	Started(+ date), completed(+ date), stopped(+ date)	The start event indicates that the end user (consumer) start receiving the media content. The completed event must be sent to the tracker when the user finishes receiving the media content. The stop event sent by the end user indicates to the tracker that the client is shutting down gracefully.
Announce URL of the Tracker		
Seeders list	Seeder's identifier Seeder's IP address Seeder's listening port	List of seeders providing the same content identified by "Content identifier".

Table 11: Session Metadata Description

2.4.7 Peer Metadata

In addition to standard information related to a peer such as its ID, IP address, listening port, etc, the peer metadata class also allows to describe the peer functionalities. In this class, we describe the adaptation and caching capabilities of a peer (input/output stream formats, multicast support, caching size, peer ranking, etc). Such information is considered as mandatory in any adaptation/caching process in order to decide about the peers responsible for performing these functionalities.

Table 12 describes in detail the peer metadata class elements.

Metadata entity	Detail	Description
Peer_id		A unique identifier of the peer.
IP address		

Metadata entity	Detail	Description
Listening port		
Requested pieces	List of requested pieces	The media content exchanged is divided in pieces. User can ask a piece from one peer and another piece (of the same content) from second peer.
Available pieces	List of available pieces	List of pieces that the peer can offer and appropriate description (time information, content id).
State	Choked, unchoked, interested.	Chocked: The peer can't satisfy a new request. Unchoked: The peer can response a new query. Interested: In communication with a seeder, a peer can express his interest in pieces of content announced by the seeder.
Maximum number of sources wanted		Number of peers that the client would like to receive from the tracker.
Adaptation capabilities	• Input/output formats • FEC algorithm, initial block • IP multicast support (yes/no) ○ if yes, IP multicast address	
Caching capabilities	• caching algorithm, caching size, • Peer ranking	

Table 12: Peer Metadata Description

2.5 ENVISION Metadata Management

2.5.1 Metadata Workflow

Before studying the metadata management in ENVISION, first we start by looking on metadata flow through ENVISION architecture. This section focuses on the mapping of the metadata flow on the architecture studied in D2.1 [D2.1]. Figure 6 represents the functional architecture of ENVISION as defined in D2.1, on which we have mapped the different metadata flows (M0, M1 … and M7). At the network level, the block 8 ("*Network Data Management*") which the main functionalities are to process, store and provide information about the network, provides to block 4 ("*Data*

Management") metadata about the network topology, network services, access network capabilities and loads of network equipments, via the metadata interface M1. Block 4 relays, through the interface M5, this information to block 5 ("*Overlay Management*"). This later remains the major block of the architecture. It is in charge of managing the cross-domain application overlay. It implements the overlay optimisation algorithms. This block is aware of the complete content distribution context. It gathers, in addition to network metadata from block 4, information about the end user, terminal capabilities, content and session metadata from the bloc 1 ("End-user Application Management"). It receives also metadata about the available overlay services (e.g. adaptation capabilities, caching) from the block 3 ("Service control") via the interface M4. Block 6 ("Network service control") manages the available network services as multicast, caches, etc. It provides metadata about network services to Block 9 via the interface M3.

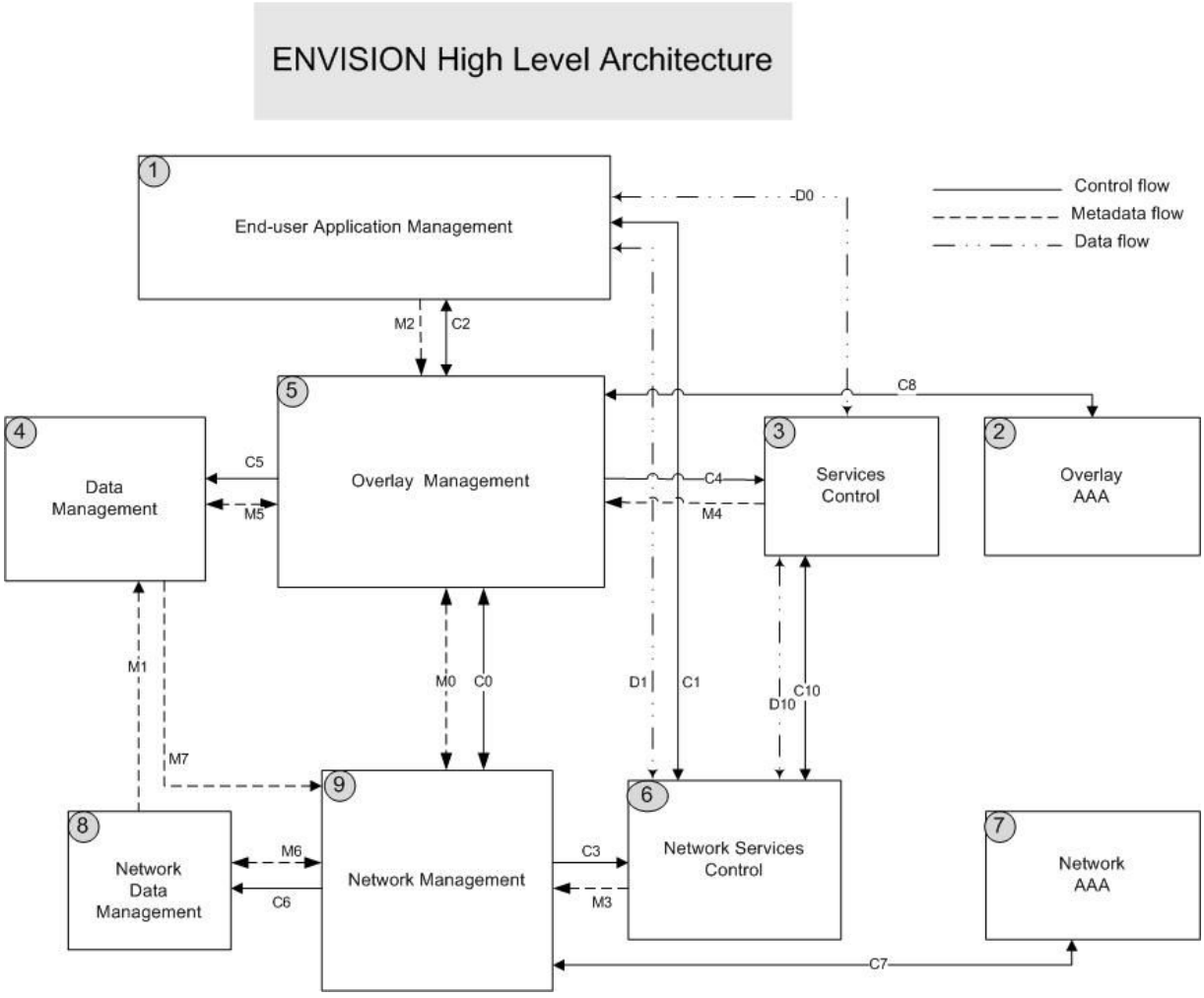


Figure 6: Metadata Flow Mapped on ENVISION Architecture

2.5.2 Metadata Modelling

Section 2.4 provides a detailed description of the metadata elements necessary for ENVISION. The aim of this section is to investigate the representation format of the metadata entities, and the modelling of the seven classes of metadata defined above.

2.5.2.1 Representation Format

The eXtensible Markup Language, XML [BPM98], has been standardised by the Worldwide Web Consortium, W3C. It is now the widely used syntax for network and content description metadata. XML users define their own markup language format, and they are not limited by a set of tags predefined by proprietary vendors. The Portability is another advantage of XML. Indeed, it is platform independent. It is supported by simple personal computer as by recent terminals such as PDA, iPhone, iPad, etc. In addition, XML allows encryption, which could be used to provide signalling security. For these reasons ENVISION metadata are represented in XML.

2.5.2.2 XSD Schema for Metadata Modelling

The XSD (Xml Schema Definition) seems to be the natural way to model XML based metadata. XSD is a recommendation of the World Wide Web Consortium (W3C) and it specifies how to formally describe elements in a XML document. It is used to express a set of rules to which an XML document must conform in order to be considered 'valid'. The XML Schema definition model includes:

- The vocabulary (element and attribute names)
- The content model (relationships and structure)
- The data types

XSD provides support for namespaces, can constrain data based on common data types, and presents object oriented features such as type derivation. Figure 7 illustrates XSD visual representation of the end user metadata class. This XSD defines the structure and different tags of the XML file representing this class:

- `General information` tag, composed of the following elements:
  - `FirstName` tag of type string.
  - `LastName` tag of type string.
  - `ContactInformation` tag describes the contact information of the user. It is of type "`ContactInformationType`", defined as a common type composed of the following fields: `Street` (String), `PostCode` (integer), and `City` (String).
  - `Photo` tag of type string. It represents the URL of the user's photo if available (optional tag)
  - `Status` tag of type string. It indicates the status of the user: simple, premier, etc.
- `AuthentificationInformation` tag
- `LocalisationInformation` tag
- `DisplayPresentationPreferences` tag
- `AudioPresentationPreferences` tag
- `UsageHistory` tag
- `AdaptationPreferences` tag
- `UserRightsOnContent` tag

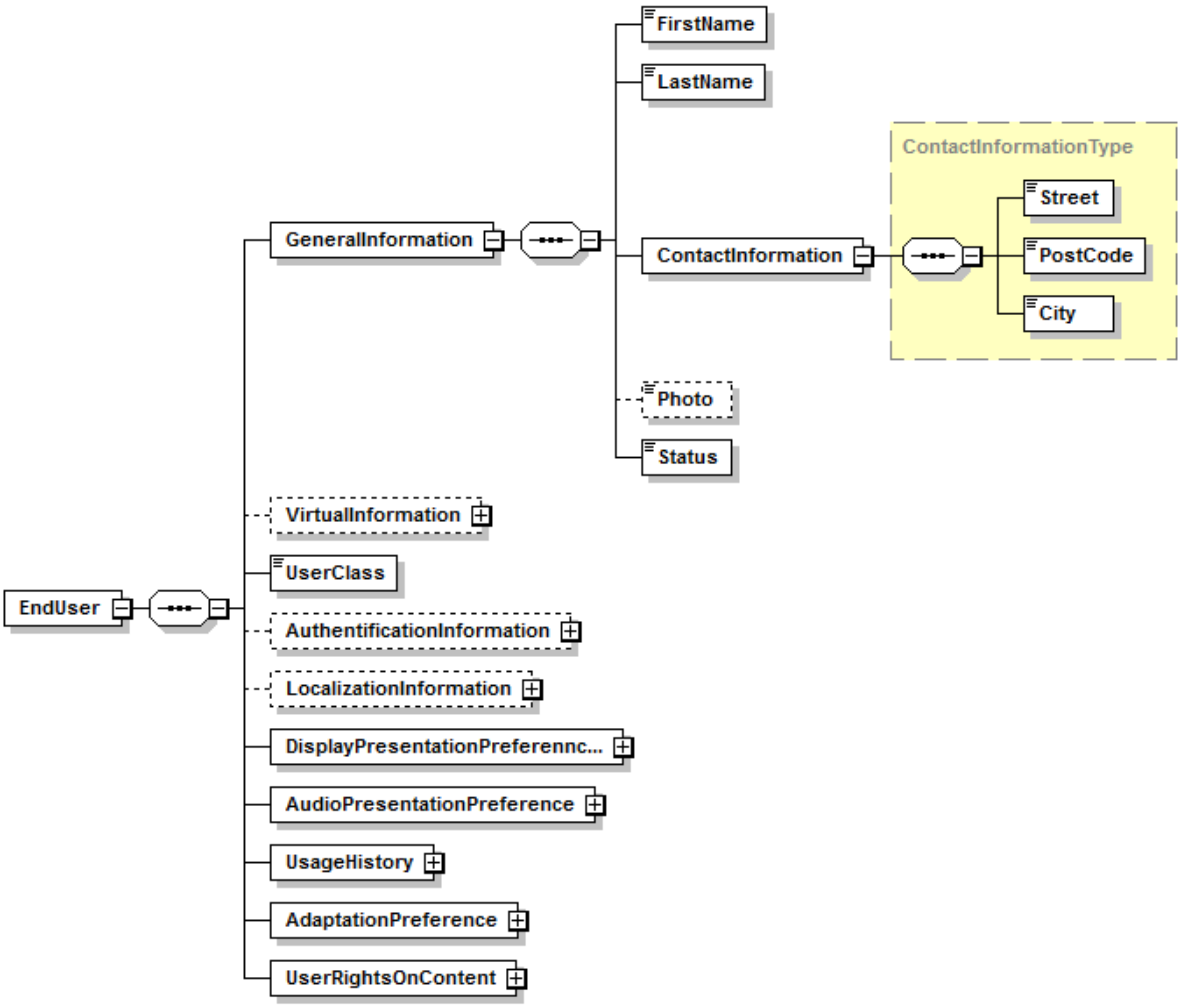
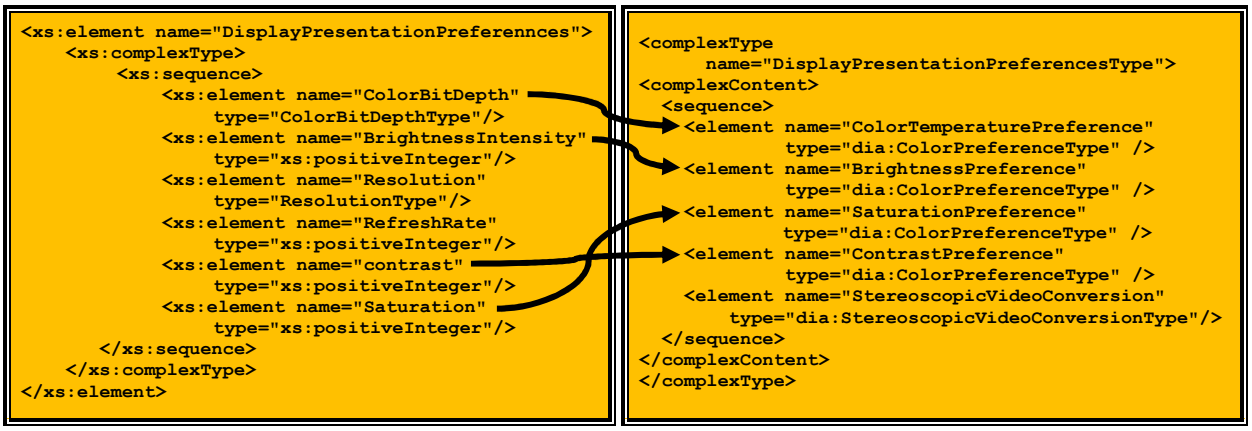


Figure 7: End User Metadata Class Modelling with XSD

The XDSs modelling the seven classes which represent ENVISION metadata are given in Appendix A.

Our format remains flexible since it can be mapped to different standards including MPEG-7, MPEG-21, TV-Anytime, etc. Indeed we can map the elements of each metadata class we defined in ENVISION on the corresponding standard element, namely to elements of the standards: MPEG-7, MPEG-21 and TV-Anytime studied in section 2.2. Figure 8 illustrates an example of mapping of some elements of the End user metadata class to standards elements. We note that in this example the property "`resolution`" member of "`displayPresentationPreferences`" is not represented in the MPEG-21 "`DisplayCapabilityBaseType`" element. For that reason we propose to extend the UED of MPEG-21 to support this property. The new structure of this element is given in Figure 9.



ENVISION Display presentation preferences metadata

MPEG-21 Display presentation preferences metadata

Figure 8: Example of ENVISION Metadata Mapping on MPEG-21 Standard

```
<!--  Definition of DisplayPresentationPreferences   -->
<!-- ################################################-->
<complexType name="DisplayPresentationPreferencesType">
    <complexContent>
        <extension base="dia:UserCharacteristicBaseType">
        <sequence>
          <element name="ColorTemperaturePreference" type="dia:ColorPreferenceType"/>
          <element name="BrightnessPreference" type="dia:ColorPreferenceType" />
          <element name="SaturationPreference" type="dia:ColorPreferenceType" />
          <element name="ContrastPreference"   type="dia:ColorPreferenceType" />
          <element name="StereoscopicVideo" type="dia:StereoscopicVideoConversionType"/>
          <element name="Resolution" type=”dia:ResolutionType” />
       </sequence>
    </complexContent>
</complexType>
```

Figure 9: Extension of “DisplayPresentationPreferencesType” Tag in MPEG-21

2.5.3 Metadata Processing

We can synthesise the integrated management of ENVISION metadata with highlighting two kinds of distinctions: first, from temporal perspective with the metadata that are dynamically conveyed along with the digital content stream and those that are statically stored in local file systems or Data Base Management System (DBMS), and from a dependencies perspective with the metadata that are content-dependent or independent or context-dependent.

Content-dependent metadata, such as *Content metadata* class, can be conveyed along within the bitstream. Other content-independent metadata can also be included as the intellectual property management and protection of the content. These are implementations made from the different available adaptation description and rights description proposed by the service and content providers and stored in local file systems or local database management systems. Conjointly, additional context-dependent metadata are collected through the ENVISION network: they include Network QoS information, terminal information, usage and environment information. They can be used for the adaptation processing. Context-dependent metadata comes from the network and the terminal device.

2.5.3.1 Metadata Gathering, Extraction and Generation

ENVISION metadata is built using metadata gathering and extraction techniques. We distinguish two kinds of metadata gathering methods: the explicit and the implicit. Explicit collect of metadata consists in interrogating an ENVISION entity (End user for example) directly on it needs, whereas, in implicit method, the profile is built by examining (in real time or not) the concerned entity behaviour.

This information must be analysed and classified according to the type of data viewed (consulted documents, clicks, bought products either wanted, Web search, etc.), before it is stored in a database. This perpetual enrichment permits to have a unified view of user behaviour and network operating at anytime. Details about these two kinds of *gathering* are described in what follows:

1) *Explicit metadata gathering*: In the explicit metadata collect method, the user profile is built according to the information given voluntarily by the user at the time of his subscription. The user describes, in explicit manner, his personal information and his centres of interests by filling in a form for example. This is why it is called declarative or static modelling of user preferences.

   Terminal capabilities including hardware properties such as processor speed, software properties such as operating system and display properties such as screen resolution are classified as explicit metadata since they are gathered in explicit manner by interrogating the terminal information registry.

   The explicit metadata gathering method is adopted to build the network capabilities metadata, too. The maximum bandwidth offered by the network, the maximum numbers of simultaneous active sessions, the ISP preferences/rank, etc. are obtained as a response to an explicit request to the ISP.

2) *Implicit metadata gathering*: In the implicit metadata collect, the profile content is built according to users and network's running and behaviour. The history of received streams, electronic subscriptions, demands of videos are, as many others opportunities, a source to collect some information on user. The average measured end-to-end delay, jitter, and consumed bandwidth and the packet loss are implicitly deducted from the network. The implicit personalisation provides dynamic (evolutionary and extensible) modelling of user preferences and the network behaviour. The implicit user personalisation presents the advantage that it does not require an active involvement of the user.

The combination of the explicit and implicit metadata gathering techniques proves to be more efficient when integrating training techniques or the intelligent agents. These techniques put up to date or revalue the user profile according to his actions.

2.5.3.2 Metadata Delivery

2.5.3.2.1 Metadata Delivery

In ENVISION the metadata delivery protocol is implemented as a common transport protocol. There are a several metadata delivery modes such as unicast or multicast, subscribe-notify, query-response or a mixture of those styles. These methods can be implemented either as a web service based platform or as a SIP-based service or as an extension to the P2P protocol. The following points must be decided when specifying the delivery protocol:

- Push or pull mode of delivery
- Unicast or multicast, ensuring reliability or not
- Query-response; metadata bi-directional transport, XQuery etc.

2.5.3.2.1.1 Metadata General Delivery Framework

The delivery process of ENVISION metadata can be viewed as the result of five distinct processes: Metadata can be extracted or generated in the several ways:

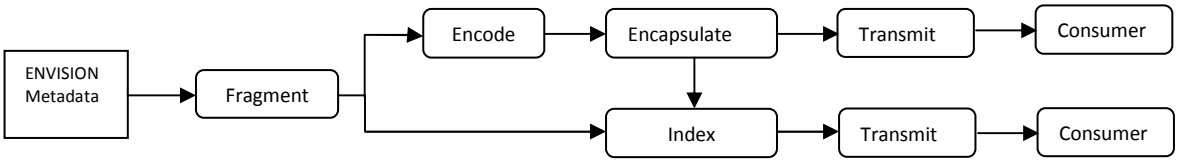


Figure 10: Metadata Delivery Process

2.5.3.2.1.1.1 Fragmentation

To enable the efficient delivery, updating and navigation of an ENVISION metadata description, a number of fragment types have been defined. A fragment is the ultimate atomic part of an ENVISION metadata description that can be transmitted independently to an ENVISION entity. A fragment shall be self consistent in the sense that:

It shall be capable of being updated independently from other fragments.

The way it is decoded, processed and accessed shall be independent from the order in which it is transmitted relative to other fragments.

Fragmentation process allows updating only metadata fragment which has been changed without need to send back all the content of metadata class. For example, in Figure 11, when end user localisation information changes and needs to be updated, only the fragment conveying this information (fragment B) will be sent back and there is no need to send back all the end user metadata information.

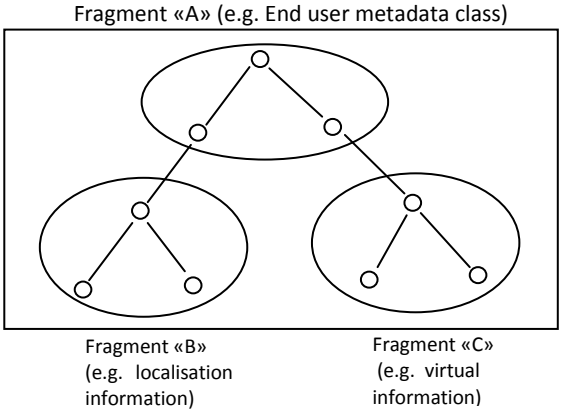


Figure 11: Metadata Fragmentation

2.5.3.2.1.1.2 (Void) Encoding

(Void)

2.5.3.2.1.1.3 Encapsulation

Once the fragments have been encoded they need to be *encapsulated*. The process of encapsulation provides further information to enable a receiving device to manage a set of transmitted fragments. A receiver needs to be able to uniquely identify a fragment within the ENVISION metadata fragment stream and also to be able to identify when the data within a fragment changes. This information is provided by the encapsulation layer.

For the transmission of fragments, the encapsulation mechanism shall be used.

2.5.3.2.1.1.4 Indexing

Within an ENVISION metadata fragment stream there are likely to be many hundreds of fragments. Due to the volume of information necessary to provide the enhanced functionality expected of a user it is important that there is an efficient mechanism for locating information from within the ENVISION metadata fragment stream. In addition to enable a device to quickly find a fragment of interest, indices can also for example be used to provide enhanced functionality such as an A-Z listing for Content Titles, Genre Listing etc. Indexing is an optional part of the present document, however it is seen as a powerful mechanism when ENVISION metadata is to be delivered to receivers that have limited processing and storage capabilities.

2.5.3.2.1.1.5 (Void) Metadata transport

(Void)

2.5.3.2.1.2 Metadata Updating Method

Out-dated metadata should be updated if delivered metadata is already cached or stored in the metadata client. The versioning and updating for delivered metadata is managed through a fragment updating method. The metadata consistency and reliability must be guaranteed by this method itself. When changes in the original metadata instance occur, it should be possible to notify clients with replicated instances of this metadata. A mechanism should also be provided for maintaining the atomicity, consistency, isolation and durability of the related Metadata Fragments.

If the Metadata source detects a modification in metadata instances already delivered, the most current version of those instances must be propagated to the clients and outdated metadata instances should be automatically discarded and updated to the latest version. Notification can be announced at a scheduled time or on demand by a client request.

2.5.3.2.2 (Void) Metadata Storage and Access Control

(Void)

3. CONTENT GENERATION

3.1 Introduction

Content generation in the scope of the ENVISION project refers to the encoding methods and representation of the input video stream in a compressed manner to fit with the underlying transport available on heterogeneous access networks. In terms of numbers, it means to reduce a 1.15 Gbps (HD format) or 200Mbps (SD format) raw stream into 100k and up to 2-3Mbps at most. In this section we provide some background information on video encoding standards and methods that are candidate to be part of the ENVISION solution, we examine the current trends in content compression and propose a design for content generation. We mainly investigate advanced video compression format such as H.264 SVC profile which allows to encode the video once and to decode it many-time. The main reason for that is that SVC allows larger sharing of generated content between different users, which saves storage space, saves encoding processing, and allows more efficient transportation over multicast trees and branches.

3.2 State of the Art

The ITU-T and the ISO/IEC JTC1 are the two organisations that develop well-established video coding standards. The ITU-T video coding standards are denoted with H.26X (e.g. H.261, H262, H.263 and H.264). The ISO/IEC standards are denoted with MPEG-x (e.g. MPEG-1, MPEG-2 and MPEG-4).

The ITU-T standards have been designed essentially for real-time applications, such as video conferencing, while the MPEG standards have been designed mostly to address the needs of video storage (DVD), broadcast video and video streaming applications. For the most part, the two standardisations committees have worked independently on the different standards. The exceptions are H.262/MPEG-2, completed in 1994, and H.264 (also called MPEG-4 Part 10 or MPEG-4 AVC) finalised in 2003 and H.264/SVC completed in 2007. The last addition to the specification was the multiview coding extension H.264/MVC in early 2009. Currently, the JVT (Joint Video Team between ITU-T and ISO/IEC) is working on a new standard named High Efficiency Video Coding (HEVC) or H.265, expected in 2013. Figure 12 illustrates the evolution of different A/V standards driven by ITU-T and the ISO/IEC.

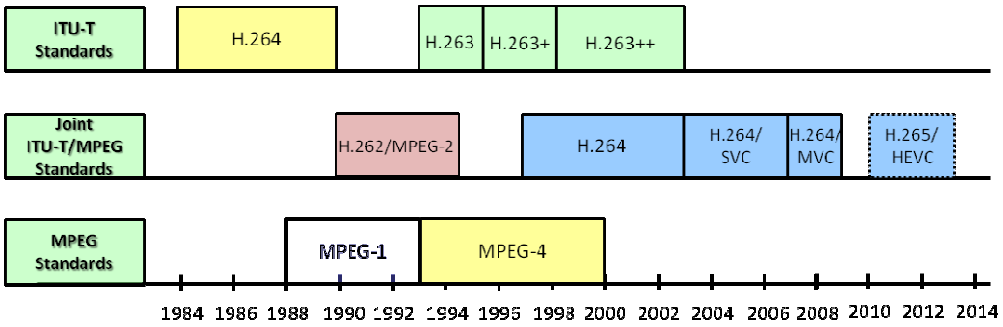


Figure 12: Progression of the ITU-T Recommendations and MPEG Standards

In this section we present the most important video coding standards namely MPEG-2, MPEG-4, H264/AVC and H.264/SVC along with the recent emerging standards such as VP8 and VC-1.

3.2.1 MPEG-1 and MPEG-2

MPEG is an encoding and compression system for digital multimedia content defined by the Motion Pictures Expert Group. MPEG-2 [HPN97] extends the basic MPEG system to provide compression support for TV quality transmission of digital TV pictures. Since the MPEG-2 standard provides good

compression using standard algorithms, it has become the standard for extensive and continuously growing systems: Digital TV (cable, satellite and terrestrial broadcast), Video on Demand (VoD), Digital Versatile Disc (DVD), personal computing, etc.

MPEG-1 and MPEG-2 offer different compression rates: MPEG-1 provides good quality (TV-like) up to 3 Mbps, MPEG-2 provides a compression solution for distribution that can provide bandwidth of 3 to 15 Mbit/s. MPEG-2 4:2:2 profile is the selected compression profile for high quality contribution applications, with bit rates up to 50Mbps.

It is important to note that in order to achieve TV-like quality transmission over a low and variable channel bitrate such as Internet delivery, MPEG-1 and MPEG-2 are not appropriate. For these applications, using MPEG-4 or H.264 would enhance considerably the quality of the delivered service.

3.2.2 MPEG-4

The main objective of MPEG-4 [PE02][PE02] is to provide a standard representation supporting different ways of communications, various services scenarios (Broadcast, Communication, Retrieval) and several delivery technologies taking into account also QoS mechanisms. The most innovative concept in MPEG-4 part 2 is the object-based representation approach where an audiovisual scene is coded as a composition of natural and synthetic objects. Indeed in MPEG-4, each Elementary Stream can contain different types of information, audiovisual (AV) object data, scene description information, and control information in the form of object descriptors.

Mainly MPEG-4 architecture is composed of three layers (Figure 13): compression, synchronisation and delivery layers:

- The compression layer performs media encoding and decoding of Elementary Streams.
- The synchronisation layer manages Elementary Streams, their synchronisation and hierarchical relationship. It allows the inclusion of timing, fragmentation, and continuity information on associated data packets.
- The delivery layer ensures transparent access to content irrespective of the delivery technology. The MPEG-4 data can use different transport protocols:
  - MPEG-2 Transport Streams (MPEG-2 TS)
  - UDP (User Datagram Protocol) over IP and RTP (real-time Transport Protocol)
  - ATM AAL2
  - MPEG-4 (MP4) files.

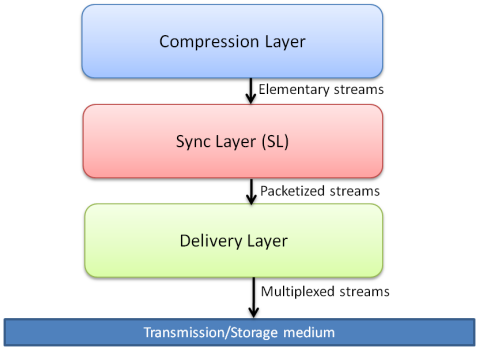


Figure 13: MPEG-4 General Structure

3.2.3 H.264

Since 1997, the IUT-T video experts Group (VCEG) has been working on new coding standard, namely H.26L. In late 2001, MPEG video group and VCEG decided to work together as Joint Video Team (JVT) to create a single technical design for forthcoming ITU-T Recommendation and for new part of ISO/IEC MPEG-4 standard. The final working version is called H.264 AVC (Advanced Video Coding) [R10] and ISO/IEC 14496-10 (MPEG-4 part 10). It has been adopted in July 2003 as a standard in Berlin Meeting.

The codec specification itself distinguishes conceptually between a video coding layer (VCL), and a network abstraction layer (NAL). The VCL contains the signal processing functionality of the codec, things such as transform, quantisation, motion search/compensation, and the loop filter. It follows the general concept of most of today's video codecs, a macroblock-based coder that utilises inter picture prediction with motion compensation, and transform coding of the residual signal. This standard enables higher quality video coding by supporting increased sample bit depth precision and higher-resolution colour information, including sampling structures. Furthermore, it also provides valuable error-resilient support for delivery the media content.

The basic configuration of the H.264 codec is similar to H.263 and MPEG-4 (Part 2). The image width and height of the source data are restricted to be multiples of 16. Pictures are divided into macroblocks of 16x16 pixels (8x8 and 4x4 is also support for high profiles). A number of consecutive macroblocks in coding order can be organised in slices. Slices represent independent coding units in a way that they can be decoded without referencing other slices of the same frame. The outputs of the VCL are slices. The NAL encapsulates the slices into Network Abstraction Layer Units (NALUs) which are suitable for the transmission over packet networks or the use in packet oriented multiplex environments [ABI01].

The H.264 achieves 50% coding gain over MPEG-2, 47% coding gain over H.263 baseline, and 24% coding gain over H.263 high profile encoders within the motion compensation loop in order to reduce visual artefacts and improve prediction [KA03].

One of the main properties of the H.264 codec is the complete decoupling of the transmission time, the decoding time, and the sampling or presentation time of slices and pictures. The codec itself is unaware of time, and does not carry information such as the number of skipped frames (as common in the form of the Temporal Reference in earlier video compression standards). Also, there are NAL units that are affecting many pictures and are, hence, inherently time-less. The IETF AVT group has defined RTP payload format for H.264 codec, which defines essentially timing information of NAL units [KA03].

Conceptually, the design of H.264/AVC covers a *Video Coding Layer* (VCL) and a *Network Abstraction Layer* (NAL). While the VCL creates a coded representation of the source content, the NAL formats these data and provides header information in a way that enables simple and effective customisation of the use of VCL data for a broad variety of systems.

3.2.3.1 Video Coding Layer (VCL)

The VCL of H.264/AVC follows the so-called block-based hybrid video coding approach. The way pictures are partitioned into smaller coding units in H.264/AVC, follows the traditional concept of subdivision into *macroblocks* and *slices*. Each picture is partitioned into macroblocks that each covers a rectangular area of 16x16 luma samples. In the case of the 4:2:0 chroma sampling, the macroblocks for chroma samples are 8x8 large. The samples of a macroblock are either spatially or temporally predicted. The macroblocks of a picture are organised in slices, each of which can be parsed independently of other slices in a picture. A picture in H.264/AVC can be a collection of one or several slices. H.264/AVC supports three basic slice coding types:

- I-slice: *intra-picture* predictive coding using spatial prediction from neighbouring regions

- P-slice: intra-picture predictive coding and inter-picture *predictive* coding with one prediction signal for each predicted region,
- B-slice: intra-picture predictive coding, inter-picture predictive coding, and inter-picture *bi-predictive* coding with two prediction signals that are combined with a weighted average to form the region prediction.

Figure 14 below shows a representation of slice I, P and B within a H.264/AVC stream.

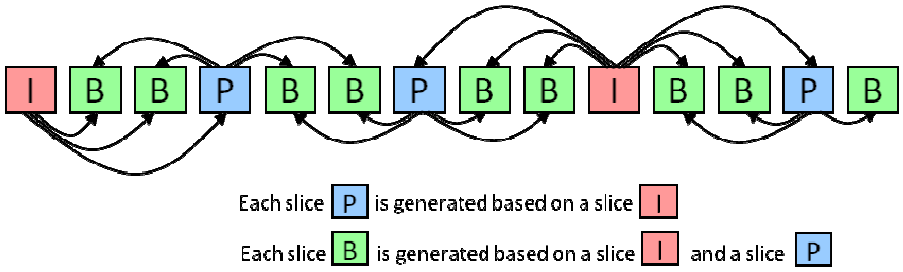


Figure 14: Slice I, P and B within a H.264/AVC Stream.

3.2.3.2 Network Abstraction Layer (NAL)

The coded video data are organised into NAL units. NAL units are packets that contain an integer number of bytes. A NAL unit starts with a one-byte header, which signals the type of the contained data. The remaining bytes represent payload data. NAL units are classified into VCL NAL units, which contain coded slices or coded slice data partitions, and non-VCL NAL units, which contain associated additional decoding information and signalling for the codec. The most important non-VCL NAL units are *parameter sets* and *supplemental enhancement information* (SEI). The *sequence parameter sets* (SPS) and *picture parameter sets* (PPS) contain infrequently changing information for a video sequence. SEI messages are not required for decoding the samples of a video sequence. They provide additional information which can assist the decoding process or related processes like bit stream manipulation or display. A set of consecutive NAL units with specific properties is referred to as an *access unit* (AU). The decoding of an access unit results in exactly one decoded picture. A set of consecutive access units with certain properties is referred to as a *coded video sequence* (CVS). A CVS represents an independently decodable part of a NAL unit bit stream. It always starts with an *Instantaneous Decoding Refresh* (IDR) access unit, which signals that the IDR access unit and all following access units can be decoded without decoding any previous pictures of the bit stream.

3.2.4 Scalable Video Coding (SVC): H.264/SVC

Scalable video coding (H.264 SVC) is a set of new scalable extensions for H.264 standard that is considered the most promising video format for media streaming over heterogeneous networks [CHG05] [SMW04] [SMW06]. Specified in Annex G of H.264/AVC, SVC allows the construction of bitstreams that contain sub-bitstreams that can be consumed by heterogeneous clients. A scalable video coding is capable to produce highly compressed bitstreams, in order to create a wide variety of bitrates.

In SVC encoding scheme, each video stream is encoded in multiple video quality layers. Each layer can be decoded to provide different video characteristics. The first layer provides the basic quality of the video is called “Base Layer” while other layers, which are used to enhance the overall video quality of the base layer, are called “Enhancement Layers” [SVX07] [WHZ00].

An original SVC stream can be truncated to produce video of different qualities, sizes, and frame rates, i.e. in SNR, spatial and temporal dimensions. This scalability makes SVC bitstreams suitable for

heterogeneous networks and terminals to meet the QoS requirements restrictions often encountered by the streaming applications. In SVC stream, the base layer is encoded using a fully standard compatible H.264 AVC (Advanced Video Coding). Then enhancement layers can be added, each providing temporal, spatial, or SNR scalability. The SVC format has the ability to provide the decoder with different enhancement layers depending on reception of the base layer and lower enhancement layers. Thus a transmission scheme should ensure that these layers are transmitted such that packet loss is kept as low as possible even for high overall transport packet loss rates. Besides the ability to adapt to different heterogeneous networks and terminals, SVC can also achieve graceful degradation of video quality in case of packet loss and high end-to-end delay, as the decoder will successfully decode the stream even in the absence or late arrival of some layers.

The objective of the SVC standardisation has been to enable the encoding of a high-quality video bitstream that contains one or more subset bitstreams. Those sub-streams can themselves be decoded with a complexity and reconstruction quality similar to that achieved using the existing H.264/AVC design with the same quantity of data as in the subset bitstream. Hence, it enables the transmission and decoding of partial bitstreams to provide video services with lower temporal or spatial resolutions or reduced fidelity.

3.2.4.1 Types of scalability in SVC

The term “scalability” refers to the removal of parts of the video bitstream in order to adapt it to the various needs or preferences of end users as well as to varying terminal capabilities or network conditions. Apart from the required support of all common types of scalability, the most important design criteria for a successful scalable video coding standard are coding efficiency and complexity. Since SVC was developed as an extension of H.264/AVC with its entire well-designed core coding tools being inherited, one of the design principles of SVC was that new tools should only be added if necessary for efficiently supporting the required types of scalability.

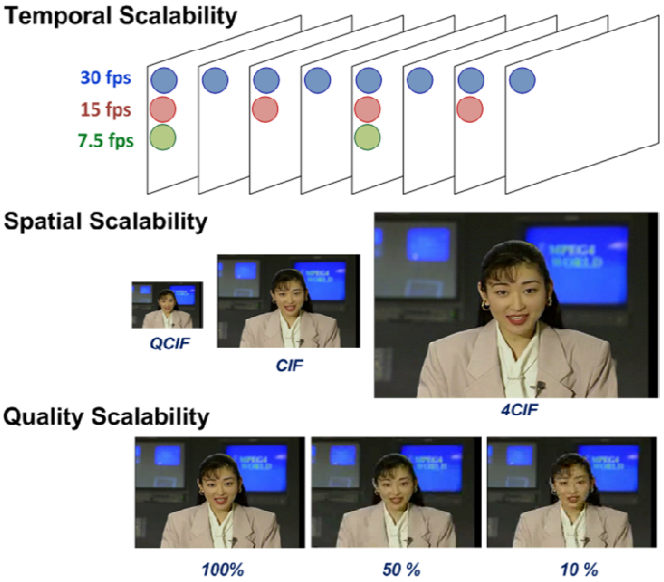


Figure 15: The Basic Types of Scalability in Video Coding

Spatial scalability and temporal scalability describe scalability approaches in which subsets of the bit stream represent the source content with a reduced picture size (spatial resolution) or frame rate (temporal resolution), respectively. With quality scalability, the sub-stream provides the same spatio-temporal resolution as the complete bit stream, but with a lower fidelity – where fidelity is often informally referred to as signal-to-noise ratio (SNR). Quality scalability is also commonly referred to as fidelity or SNR scalability. The different types of scalability can also be combined, so that a multitude of representations with different spatio-temporal resolutions and bitrates can be

supported within a single scalable bit stream. The sub-stream containing a video of a certain resolution, frame rate and fidelity is identified by a certain *scalable layer id* (SLID). The scalable layer id is an integer referencing the *quality* of a sub-stream. The higher that identifier is (SLID), the better the video quality will be.

3.2.4.1.1 Temporal Scalability

A bit stream provides temporal scalability when the set of corresponding access units can be partitioned into a temporal base layer and one or more temporal enhancement layers with the following property. Let the temporal layers be identified by a temporal layer identifier *t*, which starts from 0 for the base layer and is increased by 1 from one temporal layer to the next. Then for each natural number *K*, the bit stream that is obtained by removing all access units of all temporal layers with a temporal layer identifier *t* greater than *K* forms another valid bit stream for the given decoder.

Temporal scalability can generally be achieved by restricting motion-compensated prediction to reference pictures with a temporal layer identifier that is less than or equal to the temporal layer identifier of the picture to be predicted. The prior video coding standards MPEG-1, H.262/MPEG-2 Video, H.263, and MPEG-4 Visual all support temporal scalability to some degree. H.264/AVC provides a significantly increased flexibility for temporal scalability because of its reference picture memory control. It allows the coding of picture sequences with arbitrary temporal dependencies, which are only restricted by the maximum usable Decoded Picture Buffer (DPB) size. Hence, for supporting temporal scalability with a reasonable number of temporal layers, no changes to the design of H.264/AVC were required. The only related change in SVC refers to the signalling of temporal layers.

Figure 16 presents different hierarchical prediction structures. In this figure, the enhancement layer pictures are typically coded as B pictures. But temporal coding structure of the figure can also be realised using P slices since each temporal layer chosen can be decoded independently of all layers with a temporal layer with identifier inferior to the chosen one. The *Group of Picture* (GOP) is defined as the set of pictures between two successive pictures of temporal base layer. This prediction structure with hierarchical B or P pictures also shows excellent coding efficiency.

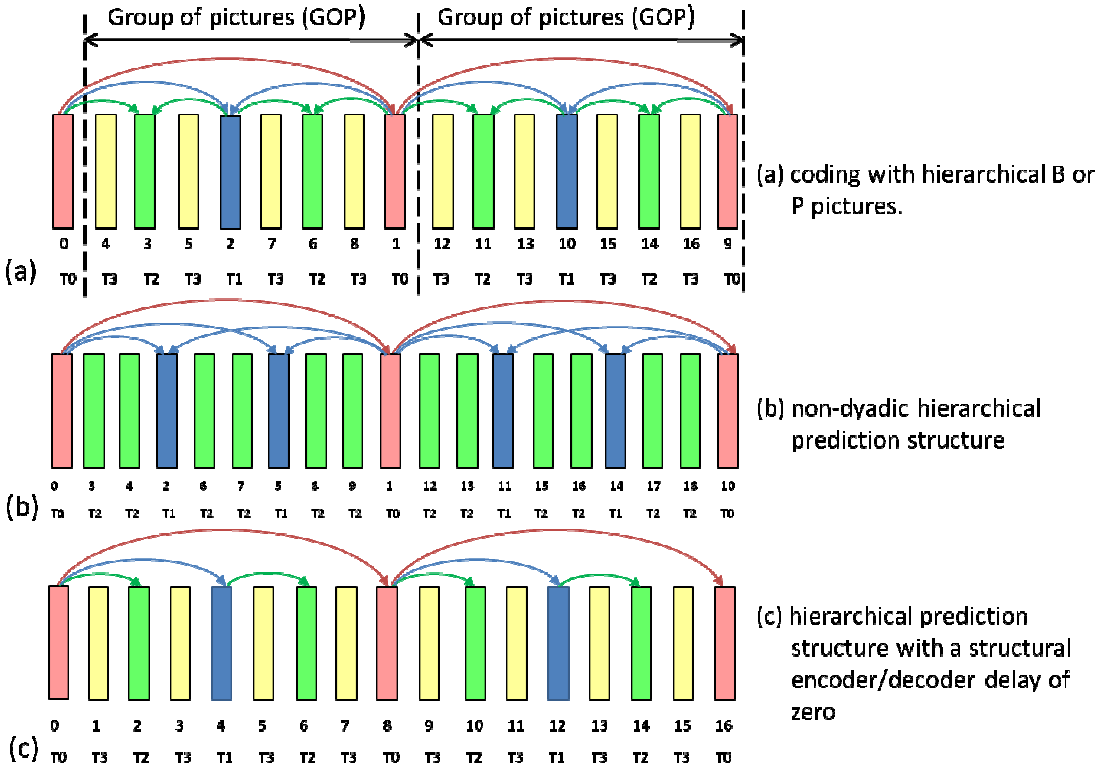


Figure 16: Hierarchical Prediction Structures for Enabling Temporal Scalability

The numbers directly below the pictures specifies the coding order. The symbols T_k specify the temporal layers with k representing the corresponding temporal layer identifier.

Furthermore, hierarchical prediction structures are not restricted to the dyadic case. As an example, Figure 16.b illustrates a non-dyadic hierarchical prediction structure, which provides 2 independently decodable sub-sequences with $1/9^{th}$ and $1/3^{rd}$ of the full frame rate. It is further possible to arbitrarily adjust the structural delay between encoding and decoding a picture by restricting motion-compensated prediction from pictures that follow the picture to be predicted in display order. As an example, Figure 16.c shows a hierarchical prediction structure, which does not employ motion-compensated prediction from pictures in the future. Although this structure provides the same degree of temporal scalability as the prediction structure of Figure 16.a, its structural delay is equal to 0 pictures, compared to 7 pictures for the prediction structure in Figure 16.a.

3.2.4.1.2 Spatial Scalability

For supporting spatial scalable coding, SVC follows the conventional approach of multi-layer coding, which is also used in H.262/MPEG-2 Video, H.263, and MPEG-4 Visual. Each layer corresponds to a supported spatial resolution and is referred to by a spatial layer or *dependency identifier* D. The dependency identifier for the base layer is equal to 0, and it is increased by 1 from one spatial layer to the next. In each spatial layer, motion-compensated prediction and intra-prediction are employed as for single-layer coding. But in order to improve coding efficiency in comparison to simulcasting different spatial resolutions, additional so-called *inter-layer* prediction mechanisms are incorporated as illustrated in Figure 17. *Inter-layer* prediction allows an exploitation of the statistical dependencies between different layers for improving the coding efficiency (reducing the bit rate) of enhancement layers.

In order to restrict the memory requirements and decoder complexity, SVC specifies that the same coding order is used for all supported spatial layers. The representations with different spatial resolutions for a given time instant form an access unit and have to be transmitted successively in increasing order of their corresponding spatial layer identifiers. But as illustrated in Figure 17, lower layer pictures is not necessarily present in all access units, which makes it possible to combine temporal and spatial scalability.

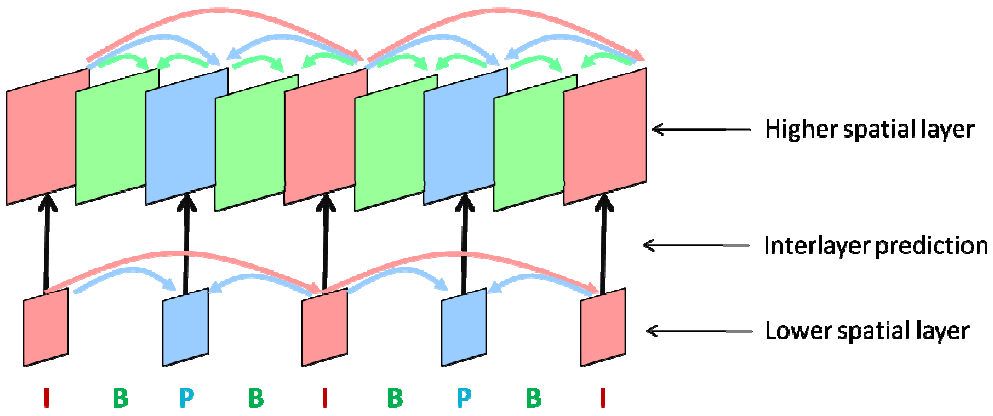


Figure 17: Multi-Layer Structure with Additional Inter-Layer Prediction.

3.2.4.1.3 Quality Scalability

Quality scalability can be considered as a special case of spatial scalability with identical picture sizes for base and enhancement layer. This case, which is also referred to as *Coarse-Grain quality Scalable* coding (CGS), is supported by the general concept for spatial scalable coding as described above. The same inter-layer prediction mechanisms for spatial scalable coding is employed, but without using

the corresponding up-sampling operations. When utilising inter-layer prediction, a refinement of texture information is typically achieved by re-quantising the residual texture signal in the enhancement layer with a smaller quantisation step size relative to that used for the preceding CGS layer.

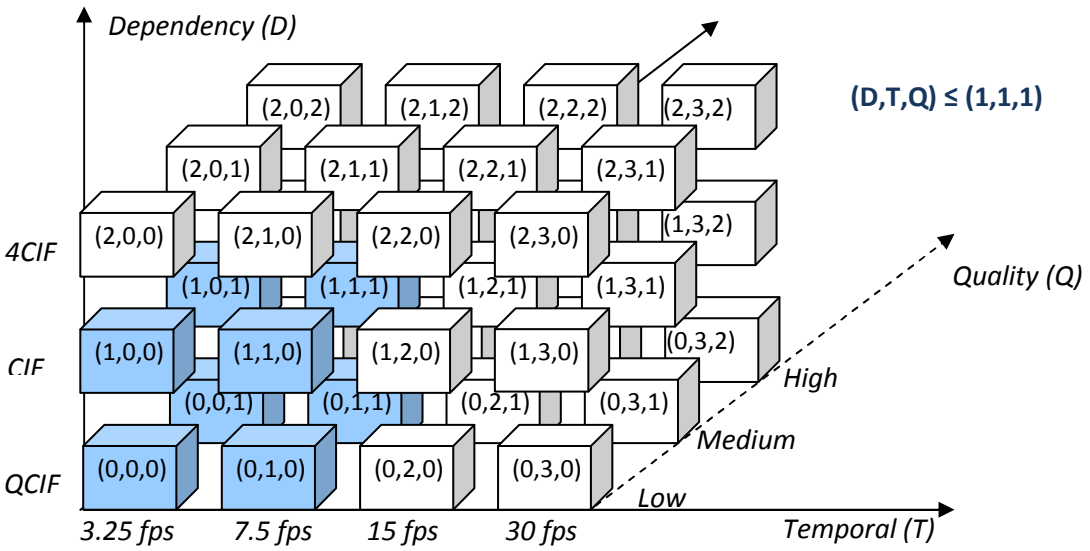
The CGS concept only allows a few selected bit rates to be supported in a scalable bit stream. In general, the number of supported rate points is identical to the number of layers. Switching between different CGS layers can only be done at defined points in the bit stream. The SVC design includes a variation of the CGS approach, which is also referred to as *Medium-Grain quality Scalability* (MGS). The MGS increases the flexibility of bit stream adaptation and error robustness, and also improves coding efficiency for bit streams that have to provide a variety of bit rates. The main improvement in MGS compared to CGS is its high-level signalling, which allows a switching between different MGS layers in any access unit. Its so-called key picture concept, which allows the adjustment of a suitable trade-off between drift and enhancement layer coding efficiency for hierarchical prediction structures.

3.2.4.2 SVC BitStream

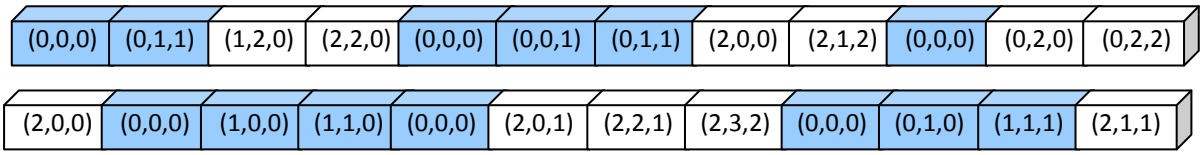
This sub-section aims to explain how SVC bitstream is structured and how to extract a sub-stream from a more complete bit stream. It will also detail the most important notions to keep in mind for switching into sub-streams with different video qualities to perform an adaptation.

3.2.4.2.1 Description of SVC BitStream

In order to achieve efficient layered extraction, scalable video bitstream consists of packets of data called NAL. A NAL represents the smallest entity that can be added or removed from the bitstream. Following such structure, an extractor simply discards NALs from the bitstream that are not needed for obtaining of video of lower quality. For easier interpretation of the extraction process, the bitstream can be represented in a 3D space as shown in Figure 18.a. The coordinates D, T and Q are respectively the number of refinement layers in Dependency (Spatial), Temporal and Quality (SNR) domain. The base layer belongs to all domains and cannot be removed from any sub-stream. We denote it as the 0^{th} layer.



a) 3D representation of a scalable bitstream



b) Linear representation of a scalable bitstream.

Figure 18: Representation of NALs within a SVC Bitstream (D,T,Q) ≤ (1,1,1).

In the example shown on Figure 18.a, we have 3 spatial layers, 4 temporal layers and 3 quality layers. Each NAL has its coordinates in the *d*-*t*-*q* space, which are denoted by (d, *t*, *q*). If (*i*, *j*, *k*) represents the desired spatial resolution, frame rate and quality of the sub-stream video, where $i \in \{0, 1, \ldots, D\}$, $j \in \{0, 1, \ldots, T\}$ and $k \in \{0, 1, \ldots, Q\}$, then, in the extraction process, the NALs with coordinates $d > i$, $t > j$, $q > k$ are simply discarded from the bitstream. In Figure 18.a, the NALs that are highlighted are the ones remaining in the final bitstream after the extraction process for which $i = j = k = 1$. Figure 18.b shows a linear representation of the same stream. The NALs that are kept to form the sub-stream are also highlighted on this figure. From that figure, we can easily conclude that the number of NALs to transmit for a sub-stream get lower as the wanted scalable layer id (SLID) is low.

3.2.4.2.2 SVC BitStream Switching

As mentioned above, switching between different quality refinements layers inside the same scalable layer is possible in each access unit. However, switching between different scalable layers id is only possible at IDR access units. In the SVC context, the classification of an access unit as IDR access unit generally depends on the target layer. An IDR access unit for a scalable layer id signals that the reconstruction of layers for the current and all following access units is independent of all previously transmitted access units. Thus, it is always possible to switch to the scalable layer (or to start the decoding of the scalable layer id) for which the current access unit represents an IDR access unit. But it is not required that the decoding of any other scalable layer id can be started at that point. IDR access units only provide random access points for a specific scalable layer id. For instance, when an access unit represents an IDR access unit for an enhancement layer and thus no motion-compensated prediction can be used, it is still possible to employ motion-compensated prediction in the lower layers in order to improve their coding efficiency.

Although SVC specifies switching between different dependencies layers only for well-defined points, a decoder can be implemented in a way that at least down-switching is possible in virtually any access unit. One way is to do multiple-loop decoding. That means, when decoding an enhancement layer, the pictures of the reference layers are reconstructed and stored in additional DPBs although they are not required for decoding the enhancement layer picture. But, when the transmission switches to any of the subordinate layers in an arbitrary access unit, the decoding of this layer can be continued since an additional DPB has been operated as if the corresponding layer would have been decoded for all previous access units. Such a decoder implementation requires additional processing power. For up-switching, the decoder usually has to wait for the next IDR access unit. However, similar to random access in single-layer coding, a decoder can also immediately start the decoding of all arriving NAL units by employing suitable error concealment techniques and deferring the output of enhancement layer pictures (i.e., continuing the output of lower layer reconstructions) until the reconstruction quality for the enhancement layer has stabilised (gradual decoder refresh).

3.2.5 Emerging Standards

3.2.5.1 WebM

Serving video on the web is different from traditional broadcast and offline mediums. WebM is an open, royalty-free, media file format designed for the web. WebM defines the file container structure, video and audio formats. WebM files consist of video streams compressed with the VP8 video codec. WebM is focused on addressing the unique needs of serving video on the web. Due to the emerging of the WebM a new royalty-free codec for the Web and HTML 5, MPEG Licensing Authority (MPEG-LA) has announced to make H.264 video royalty-free forever too.

The following describes the characteristics of WebM video format:

- Low computational footprint to enable playback on any device, including low-power netbooks, handhelds, tablets, etc.
- Simple container format,
- Highest quality real-time video delivery,
- Click and encode. Minimal codec profiles, sub-options; when possible, let the encoder make the tough choices.

3.2.5.2 The VP8 Codec

Like many modern video compression schemes, VP8 (initially developed by On2 technologies and now owned by Google) is based on decomposition of frames into square sub blocks of pixels, prediction of such sub blocks using previously constructed blocks, and adjustment of such predictions (as well as synthesis of unpredicted blocks) using a discrete cosine transform (DCT). In one special case, however, VP8 uses a "Walsh-Hadamard Transform" (WCT) instead of a DCT. Roughly speaking, such systems reduce data rate by exploiting the temporal and spatial coherence of most video signals. It is more efficient to specify the location of a visually similar portion of a prior frame than it is to specify pixel values. The frequency segregation provided by the DCT and WHT facilitate the exploitation of both spatial coherence in the original signal and the tolerance of the human visual system to moderate losses of fidelity in the reconstituted signal.

Unlike some similar schemes (the older MPEG formats, for example), VP8 specifies exact values for reconstructed pixels. Specifically, the specification for the DCT and WHT portions of the reconstruction does not allow for any "drift" caused by truncation of fractions. Rather, the algorithm is specified using fixed precision integer operations exclusively. This greatly facilitates the verification of the correctness of a decoder implementation as well as avoiding difficult-to-predict visual incongruities between such implementations.

3.2.5.3 VC–1

VC-1 is a video codec specification that has been standardised by the *Society of Motion Picture and Television Engineers (SMPTE)* and implemented by Microsoft as Microsoft Windows Media Video (WMV)-9.

The VC-1 codec is designed to achieve state-of-the-art compressed video quality at bit rates that may range from very low to very high. The codec can easily handle 1920 pixel × 1080 pixel presentation at 6 to 30 megabits per second (Mbps) for high-definition video. An example of very low bit rate video would be 160 pixel × 120 pixel resolution at 10 Kbps for modem applications. The basic functionality of VC-1 involves a block-based motion compensation and spatial transform scheme similar to that used in other video compression standards since MPEG-1 and H.261. However, VC-1 includes a number of innovations and optimisations that make it distinct from the basic compression scheme, resulting in excellent quality and efficiency. VC-1 Advanced Profile is also transport and container independent. This provides even greater flexibility for device manufacturers and content services.

3.2.6 Conclusion

In this section, we have presented the current trends of multimedia content format, namely MPEG-1, MPEG-2, MPEG-4, H.264/AVC and H.264/SVC but also emerging standards such as WebM, VP8 and VC-1.

MPEG-1 and MPEG-2 have been widely used as encoding standards and supported by various terminals but they present a relative coding inefficiency for media streaming over Internet. MPEG-4 and H.264/AVC bring certain improvement to video coding compared to ancient standards. Recently, new standards are being developed led by major companies (VP8 by Google, and VC-1 by Microsoft) and initial results are very promising. It is worth to note that all these standards do not support scalable encoding/decoding. Finally, the H.264/SVC format seems to be the most attractive solution to the problems identified in modern video transmission ecosystems, such as varying needs or preferences of end users as well as varying terminal capabilities or network conditions. Indeed, SVC has achieved significant improvements in coding efficiency with an increased degree of supported scalability relative to prior video coding standards.

3.3 Requirements for Content Generation

The following section describes some requirements for ENVISION content generation with respect to use cases specified in D2.1. We will focus on well established video format such as MPEG-2, H.264 AVC along with SVC (Scalable video coding). SVC promises to be more adapted for heterogeneous networks and terminals delivery as the adaptation can be performed by simply dropping of enhancement layers. Some requirements for content generation are described in what follow:

- Input format retrieved from the content source is used for content generation should support camera stream, raw data and file format.
- Generated content should be encoded with the following formats considered for ENVISION use cases (MPEG-2, H.264, and SVC).
- Generated content may be limited by either time intervals or size parts. Those parts called chunks are used to divide original content to allow distribution and independent decoding of the content it holds.
- Transmitted content over the network should hold content with similar priority levels. For example, it is not efficient to aggregate parts from different Frames (I, P, B) or layers (SVC). This will enable QoS-based services delivery.
- Generated content should best match between encoding profiles and parameters described in metadata (e.g. resolution, path bandwidth, etc.) and required bitrate assigned to achieve highest QoS.
- Generated content should be of the type of variable or constant coding.
- The system should support content encoding with no or minimal collaboration between generating peers.
- The system should support decoding of generated content at receiving peers while content generation process occurred in generating peers.
- The system should support decoding and encoding of media in real time.

3.4 ENVISION Content Generation Specification

The Content Generation in ENVISION will be described in an engine that is responsible for receiving input streams (e.g. from a capturing device) and producing an encoded output stream (encoding process or compression task). The input stream can be retrieved from heterogeneous sources such as video camera, microphone, DSP-stream, files, and will be encoded in an appropriate format. Along the capture process which is data loosely (due to analogue to digital conversion), the encoding process is data loosely too. In fact, the encoding process tries to represent the original AV content using the lesser possible bandwidth while preserving a quality above a certain threshold to meet end users requirements and to support large-scale delivery.

Figure 19 provides an overview of ENVISION Content Generation. The raw content provided by the content source is handled by ENVISION content generation engine to provide an encoded output stream. The generated output stream depends on the encoding parameters which themselves are driven by the metadata. The most common encoding parameters include output format (MPEG-2, MPEG-4, H.264, SVC, etc.), output bitrate, bitrate constraints (CBR/VBR mode), framerate, spatial resolution, quality level, etc. In the case of SVC format the parameters specify the number of layers, the supported spatial resolutions, the supported frame rates and the different quality levels, among other.

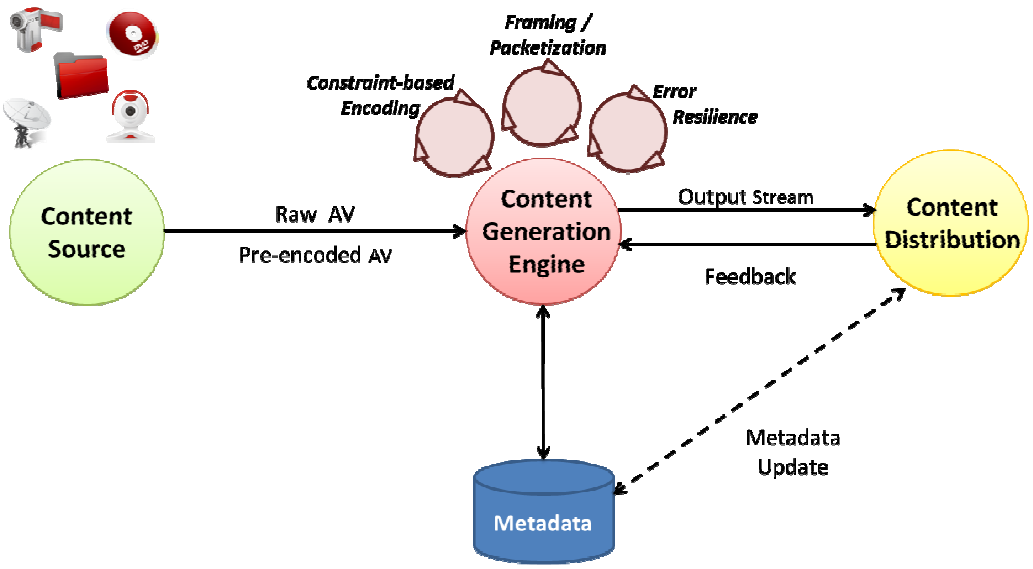


Figure 19: ENVISION Content Generation Specification

In order to decide about the content encoding parameters, the content generation engine takes into consideration the environment conditions, in particular the terminal encoding capabilities, the available bandwidth, and user's context. Terminal capabilities parameters are available in the local "*Terminal capabilities metadata"* information which includes the following parameters:

- Codec capability: Specifies formats that a terminal is able to encode to (MPEG-2, MPEG-4, H.264, SVC, etc.)
- Codec parameters: Buffers size, target bitrate, quality level, resolution, and frame rate, etc.
- Processing performance: The performance level of a terminal as determined by the particular application based on information like processor clock frequency, number of CPUs, power consumption, architecture (32/64 bits), current CPU load, etc.

- Network interface(s): In particular the bandwidth and power consumption.

In addition to terminal capabilities metadata, the content generation utilises the network metadata provided by ENVISION interface (CINA). We believe that network metadata parameters are exposed to overlay and made available to content distribution [D4.1]. Those parameters include available bandwidth, QoS mechanisms supported in the network, error correction, loss packet ratio, delay, Jitter, etc. The available bandwidth (end-to-end level) remains the most important parameter to be considered in order to adjust the output bitrate of the generated content. The challenge in using these parameters lies on construction of end-to-end knowledge and use this knowledge to achieve efficient content generation. Moreover, the feedback mechanism that can be received from content distribution (see Figure 19) allows the content generation engine to take into account in real-time the encoding parameters such as new target bitrate.

It is worth noting that end users consuming the content are heterogeneous. They may have different profiles in terms of terminals capabilities and users' preferences (e.g. display presentation preferences). Consequently, as will be investigated later, content adaptation will be used to adapt the generated content to different class of end users.

In the other hand, along the output stream, the content generation process will populate the metadata that describes the content (*Content Metadata*). In case of live streaming, some content metadata can be generated automatically in real time and conveyed within the stream, such as the content identifier, start time, content type, audio characteristics (Audio Codec, bitrate, sample frequency), video characteristics (Video Codec, resolution, bitrate, colour depth, framerate), spatial context of the content (GPS coordinate if available), etc. More elaborated content metadata can be provided explicitly by content provider depending on different streaming mechanisms (VoD, Time Shifted streaming, etc.). These metadata *are content textual description, content cost, subtitle information, Intellectual property, etc.*

Once a number of encoding parameters are gathered, the content generation must determine the best alternative among several possible operations to optimise the output format. Many methods exist to carry out this operation. We describe particularly the utility-based function which can be used as a candidate model for ENVISION. It should be noted that this optimisation is also part of content adaptation that will be described in the next section.

The utility function allows a numerical value to be assigned to one or more input parameters. The alternatives can be compared and the one with the highest utility is selected. The utility function (U) should be infinitely differentiable, monotonically increasing and additively separable.

The term monotonically increasing is described as:

$$\forall i, \frac{\partial U}{\partial x_i} \geq 0$$

Where U is the utility function with x_i can be one of the parameters: *stream bitrate, framerate, spatial resolution, etc*., for single layer codecs (e.g. H.264/AVC) or a set of quality levels, frame rates, spatial resolutions for scalable video codecs (H.264/SVC).

In other words, as those parameters increase, the utility increases and vice-versa. The utility can be seen as the QoS/QoE. Practically, the utility function can be mapped to:

1) The quality level: this constraint specifies the level of fidelity of the encoded stream to the original input stream which can be measured by different objective or subjective measurements.
2) The resulting bitrate: the necessary memory to hold the encoded stream.

Finally, the fact that the function is additively separable simplifies optimisation problems by decomposing the function. Formally it can be given as

$$\forall(i,j), U(x_i, x_j) = u_i(x_i) + u_j(x_j)$$

The influence of certain parameters in the final optimisation can be modulated by assigning to each utility function a weight indicating its relative importance.

Once the optimisation process has computed the best encoding parameters, the content will be generated accordingly. It is important to note that the content generation optimisation is invoked in the beginning as well as during the session lifetime. If those parameters were set at the beginning of the content generation, the resulting stream will be stable in terms of quality and/or bitrate. While in the second case, the resulting stream will reflect the changes made to the parameters. The main advantage of such feature is the instantaneously optimise the delivered quality.

4. CONTENT ADAPTATION

4.1 Introduction

Since the last few years, the widespread adoption of broadband residential access, availability of high bandwidth, portable and handy video capturing devices have attracted both academia and industry to propose new innovative solutions to enable content generation, sharing, and distribution among different social communities. Today's Internet is connecting millions of clients using heterogeneous terminals and through heterogeneous networks. The convergence between existing and emerging technologies such as broadband, mobile, and broadcast networks is considered as a new challenge to overcome for creating a new open and flexible platform for the delivery of media over all type of networks. In such pervasive media environments, content consumers are demanding their content to be accessible from any available home or portable devices (PC, TV, Notebook, PDA, Cellular phone, etc.) connected through different heterogeneous networks. However, to deliver the multimedia content in accordance to characteristics of distinct consumers connecting through different networks is challenging due to different constraints including available bandwidth, display capability, CPU speed, battery constraints, user preferences, etc.

Content Adaptation is the key technique that is widely used to address the above issues. Content Adaptation is a set of technologies that can be grouped under the umbrella of the Universal Multimedia Access (UMA) concept. This refers to the capability of accessing to rich multimedia content through any client terminal and network. Generally, the ability to customise/personalise any requested media content in real-time is called "adaptation". The objective of "adaptation" is to encode/modify the original video content in such a customised way that can be used "anytime" from "anywhere" (using any access network) and by "anyone" (using any terminal capability).

In ENVISION we propose a cross-layer content adaptation technique that allows multi-stream adaptation based on several dynamic profiles including those modelling network conditions and user interactions. The main idea is to support adaptation at different levels and timescales. At the application level, adaptation will be performed based on end user, terminal, network and service metadata. At the network layer, end-to-end monitoring and error resilience mechanisms will be used to enhance the reliability of adapted video streams, and forwarding data will take into account priorities assigned by the adaptation engines. In this section we presents a generalised problem formulation of content adaptation in the context of ENVISION. We point out several important and practical questions and accordingly suggested potentials methods to answer those questions.

4.2 State of the Art

The multimedia content delivery over heterogeneous networks suffers from different challenges that affect directly the perceived Quality of Service (QoS). The channel bandwidth is the first problem that impacts directly the perceived QoS. The available channel bandwidth between the receiver terminal and the content server is generally unknown and has a time-varying characteristic.

The second problem is packet loss which occurs, in general, in network element such as routers or in the access network due to channel interference and fading problems. The router queue can be overloaded by short term burst traffic leading to packet drop. Transport-level congestion control mechanism can alleviate this problem but does not avoid it. To deal with packet loss issues, content delivery applications must be designed with error control capabilities in mind.

However, all these problems can be tackled efficiently with dynamic content adaptation mechanisms. The adaptation of the multimedia content according to changing usage environments during the service delivery is becoming more and more important. That is, the characteristics of the environment where the actual multimedia content is consumed (e.g., network, terminal, and user characteristics) are varying during the consumption of the (multimedia) service. Thus, immediate actions are needed to be performed in order to enable a unique, worthwhile, and seamless

multimedia experience during the entire session lifetime. These kinds of actions are generally referred to as *dynamic multimedia content adaptation* [HAP05].

In yet another scenario where different end users are interested in the same multimedia content but with different usage environments, optimal (network) resource utilisation is achieved by transmitting the multimedia content to an intermediate node, i.e., a proxy or gateway, between the provider and the actual end users such that the offered service satisfies a set of usage environment constraints common to all end users. On receipt of the multimedia content, the proxy adapts and forwards the multimedia content satisfying the individual usage environment constraints of each end user. This kind of approach where multiple adaptation steps are successively performed within the delivery path is referred to as *distributed multimedia content adaptation* [CV05].

Furthermore, due to the emerging diversity of competing or complementary scalable coding formats, a device needs to maintain the corresponding number of hardware/software modules in order to facilitate the manipulation of multimedia content based on all these formats. This fact hampers the automatic service deployment which is one of the main goals of Next Generation Network (NGN) management. Thus, generic approaches for content handling regarding its manipulation – including adaptation – would provide promising support towards reaching this goal.

The content adaptation in ENVISION can be performed at different epochs and at different levels of service lifetime. Regarding the epoch, content adaptation can be performed during:

- The service invocation phase along with service personalisation for example used in Video on Demand (VoD) media streaming. This phase takes into consideration user profile, terminal profile, static network conditions, etc.
- The service delivery phase based on dynamic network conditions and/or feedbacks coming from the network, from the content distribution block or from the terminal (e.g. perceived quality feedback)

Regarding the levels, content adaptation can be performed as:

- Application-level adaptation (e.g. transcoding, transrating, scalable encoding, etc.) and protocol adaptation (streaming over RTP/UDP, MPEG-2 or HTTP)
- Network-level adaptation (e.g. DiffServ packet marking, re-routing etc.) which is provided by the content distribution
- Cross-Layer adaptation which performs adaptation at different level and using different parameters

An important aspect with respect to multimedia content adaptation is the actual adaptation decision-taking which aims in finding the optimal adaptation operations as a trade-off between the given multimedia content characteristics and the constraints imposed by metadata profiles in order to maximise the Quality of Service (QoS) and Quality of Experience (QoE) respectively.

In general, adaptation is necessary in the case of shortage of resources and can be performed at different levels based on the estimated end-to-end constraints. Such estimations are based on the network feedback and/or end-to-end feedback. For example, the content server (or an Adaptation Gateway) adapts its sending rate according to the available estimated bandwidth. In fact, to deal with the long-term bandwidth variation, it is necessary to choose the best codec that can generate packet at certain target rate. Short term bandwidth variation can be managed with some specific content adaptation mechanism (e.g. transrating, transcoding, etc.). Transcoding is the conversion of media content from one digital format to another format, for example from MPEG-2 to H.264.

Transrating is changing the bitrate of video stream to meet the requirements of the network or end-user device. The downgrading of the resolution, for example conversion of media content from high definition (HD) to standard definition (SD) is an example for such techniques.

Whatever the adaptation is, its main goal is to enhance the delivered video quality and to enable the terminal to access the content which was initially not designed for.

The following section describes taxonomy for video content adaptation found in literature.

4.2.1 Multimedia Content Adaptation Taxonomy

Generally, the adaptation of multimedia content is a key concept for enabling Universal Media Access (UMA) "lets viewers see anything, anywhere, anytime". It aims also to enhance the delivered video quality by maintaining the quality of service in error-prone channel. Such adaptation includes any type of data that can be transmitted over Internet including text, image, audio and video. The main advantage of the adaptation is to reduce the storage space on central content servers that no longer need to store multiple formats of original content. Furthermore, multimedia adaptation is the only way to customise/personalise the media content in real time.

In the context of content adaptation, context and content related metadata plays a significant role. Content adaptation is executed on the basis of three inputs: (1) context-related metadata (user/terminal profiles), (2) content-related metadata, and (3) adaptation capabilities supported by the content which can be described in related metadata. Context related metadata specifies the end-user preferences (content, presentation, interactions, etc.), network characteristics (available bandwidth, delay, jitter packet loss etc.) and terminal characteristics (AV capabilities, types of terminal, codecs, battery status etc.). Content related metadata specifies the media characteristics (bitrate, frame-rate, etc.) and describes the relationship between adaptation constraints. Adaptation capabilities supported by the content provides information regarding the adaptation capabilities of devices and digital rights management information determining which adaptation operations are allowed under which conditions (e.g. accessing to security key and description of bitstream map to enabled dropping of sub-stream in case of SVC). In the rest of this section, we will present a brief classification of adaptation techniques [DTH05]. In practice, different combinations of these adaptations techniques can be used depending upon the applications requirements.

4.2.1.1 Transcoding

Adaptation at signal-level or transcoding is an important technique of adaptation as earlier described. In this adaptation, an original data format is converted into another format according to the client device capability. Signal level transcoding can change the video format to another while changing several parameters: the temporal resolution (number of frames per second), the spatial resolution (the size of the image), and image quality (SNR level), for example MPEG-2 video to H.264 or WAV audio to MP3. This allows bit rate reduction, frame rate reduction, and temporal and special down sampling [WAF99]. The transcoding is applied by decoding the video to raw format (uncompressed video) and then re-encode the video to the target format. There are mainly two drawbacks when using this approach. First, the quality of the result format is generally lower or equal than the original format. Second, media transcoding generally requires extensive computation and large storage spaces, which makes this approach very expensive and cannot be scalable for large number of users. The advantage in transcoding is that it allows media adaptation both at the server and in the network. However, it can be efficient using a dedicated media gateway [AMZ95] [AMK98]. This transcoding possesses high overhead due to the potential computational complexity and may not be feasible for many real-time applications.

4.2.1.2 Semantic Event-Based Adaptation

Semantic event-based content adaptation is an active domain that takes into account the important semantic information of events present in the content. In such type of the content adaptation, content providers usually set priorities for important events or is derived by the user's preferences. For example, the point scoring event in sports videos, breaking news in broadcast services, and movement or some breaking events in the surveillance video can be marked as important that are

further used for the content adaptation. This type of adaptation is based on Metadata that are included in the video stream to allow dynamic adaptation.

4.2.1.3 Structural-Level Adaptation/Synthesis

This category of adaptation builds summaries video by changing its structure. Generally, video is composed of different events as they occur in real time. In this adaptation technique, a sequence of the key frames of original video streams is extracted that is sequentially played-out. These images can be determined by different statistical criteria. Mosaicing is another adaptation technique at structural level. This technique is very interesting in sense where an object is moving on a static background. In the mosaic image, the object is duplicated.

Synthesis adaptation is the technique that provides a more comprehensive experience or a more efficient tool for navigation. Here, the key frames of the video sequence are organised in different hierarchical structures that facilitate the browsing functions as well. The hierarchical structure can be based on temporal decomposition or semantic classification that is used for an extended view that provides an enhanced experience in comprehending the spatio-temporal relations of objects in a scene. Generally, transmission of the synthesised video stream requires much less bandwidth than the original video sequence since redundant information in the background does not have to be transmitted.

4.2.1.4 Selection/Reduction

Selection/reduction is popular adaptation approach in resource-constrained situations. This technique incorporates the selection and reduction of some elements in the video. Selection of different components of video resembles with the transcoding approach that involves the changing of bitrates or resolution of existing video stream. Reduction is the process that determines that which components of the video can be deleted while content delivery. It involves sophisticated models to determine the importance of different video components on the basis of semantic information.

4.2.1.5 Replacement

Such adaptation technique replaces the selected video elements with their less expensive counterparts while ensuring the overall value. For example, a video sequence may be replaced with still frames. The major advantage of this technique lies in terms of bandwidth efficiency because it can reduce the bandwidth requirements considerably. Moreover, this technique can also be useful in the indexing services to provide visual summaries, if bandwidth is not an issue for certain application.

4.2.1.6 Drastic temporal condensation

Drastic temporal condensation is another form of content adaptation. It is also referred as video skimming/rapid fast forward that is useful when user's preferred viewing time is very limited whereas the other resources are not very limited. This technique requires significant considerations because merely increasing the frame rate is not suitable for content adaptation.

4.2.1.7 Cross-Layer Adaptation

Cross-Layer adaptation is a new design paradigm that has been recently considered and accepted as beneficial and high performance in the research community. This model supports the interlayer communication by allowing one layer to access primitives and data information of another layer even if it is not an adjacent layer. This implies the redefinition of new design strategies more especially the interfaces offered in each layer as it breaks the classical Open Systems Interconnection (OSI) model.

The key idea in the cross-layer adaptation is to support the multimedia service at different layers of networking. At the application layer, the multimedia services will be adaptive to changing networking conditions and bandwidth availability. At the transport layer, end-to-end congestion control and

error resilience mechanisms will be used. At the network layer, resource reservation or differentiation service will be used to support end-to-end QoS provisioning techniques to provide vertical/horizontal mobility management and connectivity will be used. Feedbacks are also considered for service adaptation. Routing mechanism needs to be QoS-aware and able to handle mobility. At the data link layer Medium Access Control (MAC) needs to be modified so that reservations are respected and QoS guarantees can be supported. Adaptive power control techniques will be used to manage mobility and to maintain active links. Error control techniques will be used to protect against varying error rates. At the physical layer several options including adaptive modulation and coding are available and will be adapted to the several high level requirements. Papers [AD06] [DANB08] specify an architecture for cross-layer adaptation for wireless video on demand services. The authors describe the usage environment using MPEG-21 tools. The cross-layer adaptation is performed in a specific gateway that plays the role of an access point to support client heterogeneity and mobility. The heterogeneity of applications, terminals and networks is important aspect for cross-layer adaptation which requires more rigorous adaptation mechanisms [KPZ10]. Especially in the context of multimedia services, content adaptation is absolutely necessary due to enormous dependencies arising from heterogeneity. Cross-layer adaptation can play a key role in handling such multiplicity of dependencies [Z09]. It assists smooth transition of Internet from best effort service to QoS enabled network. The concept of cross-layer design sounds persuasively appealing but it seems a bold step after successful experience of layered architecture [KK05] [RAB02]. Right now, the research community is endeavouring following challenges in this paradigm.

- Cross-layer adaptation of the complete network infrastructure is very intricate due to handling enormous dependencies possibly in real time. A flexible architecture with proper interfacing between the layers is inevitable.
- Cross-layer design breaks the layers and hence a clean isolated implementation of different protocols is no longer possible. Each cross-layer approach affects a complete system. An analysis of these effects becomes difficult.
- The effects of coexistence of different cross-layer interactions are to be observed on system performance, maintenance and scalability. Analysis is further perplexed if different type of cross-layer optimisations are deployed across an end-to-end delivery chain. Compatibility of different cross-layer adaptation is one of the major issues. Due to lack of research in this dimension, cooperative behaviour of multiple cross-layer approaches is non-deterministic.
- Global metrics are required that maximise the utility (e.g., QoS) and minimise the cost (e.g., battery life) under various constraints by efficiently prioritising layers' local optimisation criteria.
- Optimisation of cross-layer parameters is a complex multivariate problem with various constraints derived from QoS guarantees, available bandwidth, power consumption, etc. Solutions of such optimisation problems converge in multiple iterations. Apart from the essential requirement of computational efficiency, the highly dynamic nature of wireless networks demands a rapid convergence of the solutions. Moreover, objective functions for maximum users' satisfaction have to be further investigated.
- It has to be evaluated where the actual control of a cross-layer adaptation should be located. Without a central control, different cross-layer adaptations might counteract each other. Different candidates include a separate coordinator or a particular OSI layer.
- Cross-layer adaptation simulations are generally more complex than traditional network simulations. Hybrid approaches combining network simulation tools, hardware support and analytical approaches are usually required.
- Not even a single cross-layer proposal has been tested comprehensively under real world traffic scenarios and hence QoS, power consumption and scalability of these approaches are yet to be gauged deterministically.

- The assurance of fairness is yet an un-promised reality by cross-layer design.

4.2.1.8 Other Emerging Techniques

In this section we will present a number of adaptation techniques widely used for the content adaptation beside from general classification as described above.

4.2.1.8.1 Simultaneous Store and Stream

“Simultaneous Store and Stream” also known as Simulstore [AP03] is the technique used to store different streams at the server with different spatial resolutions, temporal resolutions and SNR levels. The client connects to the server and selects the appropriate stream from a multitude of stored streams. The quality of the received video is not degraded because each stream is optimally encoded. Furthermore, this technique provides an easy selection of the appropriate stream at server side and with low complexity at server and client end. Simulstore can easily be combined with end-to-end retransmission but it has a major disadvantage that streams cannot cope with degradations in network conditions, thus, cannot be adapted to network conditions.

4.2.1.8.2 Simulcast

Simulcast means simultaneous multicasting and transmission of different streams with different spatial resolutions, temporal resolutions and SNR level. This technique enables content consumers to select an appropriate stream among the available video streams. User can switch from one stream to another to cope with dynamic network conditions.

4.2.1.8.3 Transrating

Real-time coding and transrating allows encoding the video stream right before the transmission process to meet the client requirements. A content consumer provides its specifications (End user metadata/terminal capabilities metadata for example) while connecting the content server and requested content are delivered after real-time encoding. However, real-time encoding results in a high complexity at the server and rate adaptation is not possible to serve a large number of users. Usually, this technique can be applied at an intermediate entity called “Adaptation Engine” as discussed in the next section.

4.2.1.8.4 Stream Switching

Stream switching overcomes the disadvantage of Simultaneous Store and Stream by adapting the video transmission rate to network conditions. In this technique, an active video stream is replaced with another alternative video stream, if the network conditions change (less or more network capacity) during the video transmission. However, it may lead to a synchronisation problem. To overcome this problem, streams are synchronised according to intra-coded frames (I-Frame) or special switching frame (SP-Frame). Signalling in this case is crucial for a stream switching.

With the growing proportion of connected devices (Mobile Phones, Tablets, Connected TV Sets...), it is necessary to provide the best solutions for content delivery over the Internet. Stream switching seems to be an efficient way for content adaptation and delivery and it has gained a lot of attention from leading industry companies these last two years. These industrial solutions are based on the uncontested HTTP protocol and rely on the same common principles:

- Encoding of the content in multiple qualities,
- Segmentation of the content in short segments (usually from 2 to 10 seconds),
- A "streaming" session is composed of consecutive downloads of all the segments,
- Client-driven "streaming" session (segments are pulled by the client, which dynamically and constantly choose the most appropriate bitrate)
- These solutions present some advantages such as:

- Easy compatibility with HTTP caches and CDNs,
- No NAT traversal and firewall issues,
- Fine and fast rate adaptation (Client-driven, client has the best view of its available bandwidth).

The content fragmentation, as well as the client-driven approach, makes these solutions easily compatible with P2P delivery schemes.

4.2.1.8.4.1 Apple Adaptive HTTP Streaming

Apple identified what it considers as the major issue with standard streaming, generally based on Real-Time Streaming Protocol (RTSP). The problem with RTSP is that the protocol port or its necessary ports may be blocked by routers and firewall settings, and thus preventing a device from accessing the stream. Consequently, HTTP remains an accessible standard protocol on the web. Moreover, no special server is required other than a standard HTTP server.

The basic mechanism involved in Apple HTTP streaming is to have multiple segmented MPEG2-TS files related to the same video content, but having different video quality. An ".m3u8" playlist (an extension the ".m3u" playlist specification) is sent to the client to indicate the different chunks of the content. In the case of Live Streaming, the client refreshes the playlist to see if new chunks are added to the stream. This solution introduces necessarily a minimum latency whatever segment duration is used. The protocol offers a way to specify alternate streams by pointing to a separate playlist for each alternate stream and thus performing pull-based client side HTTP adaptation. These playlists would generally be of different quality and bandwidth requirements, so the client can request the appropriate stream for whatever network conditions allow. Further, the protocol allows for the individual media clips to be encrypted so that broadcasters can limit access to paid subscribers, for instance. In this case, key files for decoding the encrypted clips are referenced in the playlist, and the client uses the key files to decrypt each one before playing. There is also a flag that broadcasters can set to disallow caching of individual media files as they are downloaded.

Apple HTTP Live Streaming specifications are available at the IETF, as an informational draft [PAN04]. This solution is implemented since iOS3 (iPhone and iPad), and also in latest version of QuickTime (on Mac OS X). It is already used by some content/service providers to deliver live and VoD content.

4.2.1.8.4.2 Microsoft Smooth Streaming

Smooth Streaming is a Microsoft technology that is part of IIS Media Services 3.0. It is enabled for streamed media from IIS to Silverlight (since version 2) and other clients (to Windows Phone 7 and recently to iOS devices as well). Using Smooth Streaming, resolutions up to 1080p can be streamed to clients and downscaled to much lower resolutions for clients with lower bandwidths. The technology works by sending the data in small fragments (around 2 seconds long) and checking that each fragment was delivered in a timely fashion. If delivery wasn't quick enough, a lower quality will be used in the next fragment. Similarly, if it was delivered very quickly, a higher quality will be tried. This variable quality is achieved through encoding the media at different quality levels. To create Smooth Streaming presentations, the same source content is encoded at several quality levels, typically with each level in its own complete file, using a compression tool. Content is delivered using a Smooth Streaming-enabled IIS origin server. Once the IIS origin server receives a request for media, it will dynamically create cacheable virtual fragments from the video files and deliver the best content possible to each end user. The benefit of this virtual fragment approach is that the content owner only needs to manage complete files rather than thousands of pre-segmented content files.

Although being a Microsoft solution, it is based on existing standards, and the specifications are publicly available. In particular, Smooth Streaming is using MPEG-4 File Format for content encapsulation with a specific feature called "movie fragments". Enabling this particular option makes it possible to store the content as an incremental set of content segments, which makes it easy for the server to extract and to send over HTTP.

It is already largely used, in particular for some major events. In France, France Television is using it to deliver live and on-demand content during some special events, such as Rolland Garros, the Tour de France or the Olympic Games. A case study of France Television experiments is presented here [FTS]

Another Microsoft solution is *Microsoft Intelligent Streaming*. It is a rather old solution and is based on MMS Streaming. In this solution, the bandwidth is constantly monitored during playback (via receiver reports), and If a bandwidth drop occurs, the server selects a lower bitrate/quality variant of the content. It is an opaque solution (no control for the service developer).

4.2.1.8.4.3 Adobe Flash Streaming Solutions

Adobe Flash dynamic streaming enables on-demand and live adaptive bit rate video streaming of standards-based MP4 media over regular HTTP connections. This capability gives content creators, developers, and publishers more choice in high-quality media delivery while maintaining the reach of the Adobe Flash Platform. While the Real Time Message Protocol (RTMP) remains the protocol of choice for lowest latency, fastest start, dynamic buffering, and stream encryption, HTTP Dynamic Streaming enables leveraging of existing caching infrastructures (for example, CDNs, ISPs, office caching, home networking), and provides tools for integrating content preparation into existing encoding workflows.

RTMP-based adaptive streaming is a feature that is available since Flash Media Server 3.5 (It also requires Flash player 9 on the client terminal). Adobe recently introduced its HTTP counterpart, called Adobe HTTP dynamic streaming (available with Flash player 10.1).

Both delivery schemes have different constraints. In particular, the oldest Adobe Flash Dynamic Streaming requires a dedicated Flash Media Server to deliver the content, while the new Adobe HTTP Dynamic Streaming can work with a standard HTTP delivery node.

In RTMP-based Adaptive Streaming, the content is encoded in multiple qualities, which are made available on a Flash Media Server 3.5. Adobe Flash Player provides, then, some QoS/QoE information (client buffer state, number of dropped frames…) on the receiver side, which makes it possible for the client to decide when a bitrate changed is required. The stream switching mechanism is managed by the client. It is to the client to do the adaptation logic and to request for a content switching; the server reacts accordingly by switching at the next synchronisation point.

4.2.1.8.4.4 3GPP BitStream Switching

Switching is a solution standardised by 3GPP in 3GPP release 6 specifications. It relies on classical RTP (+RTSP/RTCP) Streaming delivery (Figure 20). Silent adaptation [FHK06] is performed by the server based on client RTCP feedback. The drawback of this solution is that only rate adaptation is allowed (resolution cannot be changed)

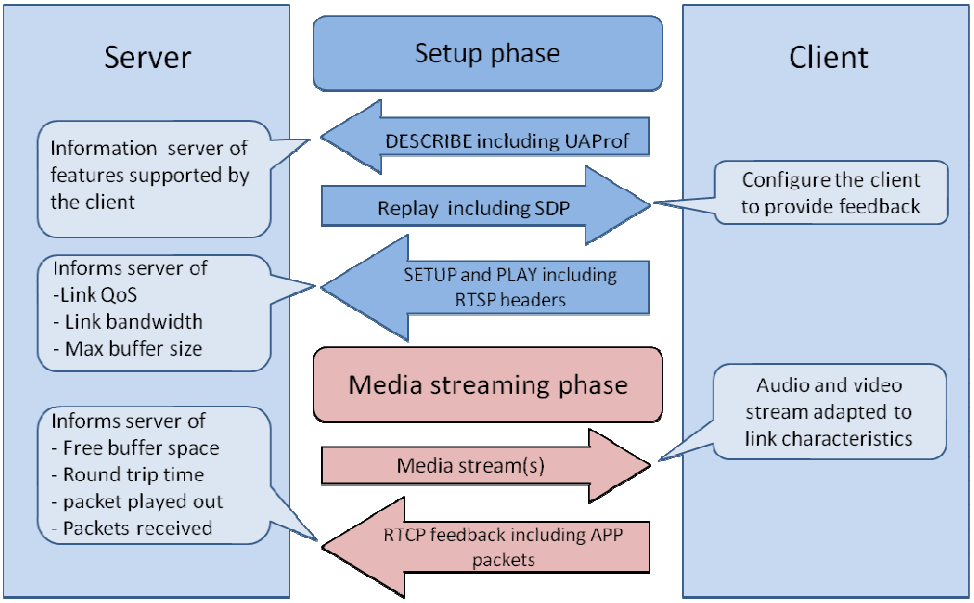


Figure 20: 3GPP Adaptive Streaming Session

4.2.2 On-going Standardisation Efforts

Following the trend on HTTP adaptive streaming, several standardisation organisations have started to work on this topic. Among these organisations:

- 3GPP: HAS (HTTP Adaptive Streaming) has been adopted in the Release 9 (2010).
- Open IPTV Forum: HTTP Adaptive Streaming has been adopted in the Release 2 (2010), based on 3GPP specifications. ([3GP07])
- MPEG: DASH (Dynamic Adaptive Streaming over HTTP) is currently being standardised, based on 3GPP specifications. Final specifications are expected for July 2011.

Those 3 Standard-Developing Organisations (SDO) exchange together for an alignment of their respective specifications. It is expected that the MPEG standard will serve as a basis for the other SDOs which will profile MPEG specifications for their particular usage.

IETF has also started some discussions on http streaming, but no consensus has been reached yet on the objectives.

The concept and general usage is the same as for existing industrial solutions. Two parts are standardised by these SDOs:

- A MPD (Media Presentation Description), and the according syntax: XML file which describes the content (resolutions, bitrates, URLs, etc.)
- The format of a segment (piece of content)

Some guidelines are also provided to describe how a set of segments can be aggregated to generate a compliant non segmented content.

The usage (how to send HTTP requests, etc.), the client logic (how to perform bandwidth adaptation) are not part of the specifications.

4.2.3 Adaptation in Some EC-Projects

This section describes the state-of-the-art regarding dynamic multimedia adaptation in EC-funded project, most notably, DANAE, DAIDALOS, aceMedia, ENTHRONE, P2P-Next, ALICANTE and COAST.

The DANAE (Dynamic and distributed Adaptation of scalable multimedia content in a context-Aware Environment, IST-1-507113) project developed and implemented, among others, a very first prototype of an MPEG-21-compliant Adaptation Node (AN) for scalable multimedia resources. The DANAE adaptation node may be located within the network and provides means for dynamic adaptation according to changing usage environments. However, this adaptation node is a prototype implementation which has not been integrated in any kind of management platform and supports only scalable multimedia resources. Hence, the appropriate support towards common Service Delivery Platforms (SDPs) and recent NGN developments is unable to facilitate the dynamic adaptation for the large scale networks. Furthermore, the AN is metadata-driven but does not provide means for adapting the metadata (i.e., content-related metadata in which the actual end user may be interested) itself.

The DAIDALOS (Designing Advanced network Interfaces for the Delivery and Administration of Location independent, Optimised personal Services, IST-2002-506997 and IST-2005-026943) project developed and implemented, among others, the Content Adaptation Node (CAN) as part of their Multimedia Service Provisioning Platform (MMSP). Although the CAN is part of the MMSP which provides management capabilities, the CAN provides only a basic means for dynamic adaptation (e.g., dynamic codec/content switching) and concentrates only on the multimedia resources. However, the dynamic network conditions are not taken into consideration while executing the adaptation.

The aceMedia (Integrating knowledge, semantics and content for user-centred intelligent media services, IST-2002-2.3.1.7) projects developed and implemented, among others, the Content Adaptation Integrator (CAIN) which provides means for multimedia resource adaptation via constraint satisfaction and optimisation taking into accounts the capabilities of the actual Content Adaptation Tools (CATs), e.g., transcoding, transmoding, and scaling. However, the CAIN is a prototype that has not been integrated in any kind of management platform and also has no support towards NGN. Furthermore, the proposed multimedia-adaptation is restricted to the semantic-based content adaptation that is capable of providing summaries of video content may not be sufficient to support new emerging video services.

The ENTHRONE (End-to-End QoS through Integrated Management of Content, Networks, and Terminal, IST-1-507637) project developed and implemented the ENTHRONE Integrated Management Supervisor (EIMS) which provides a management plan for heterogeneous (access) networks and terminals enabling end-to-end QoS for multimedia resource consumption. For adaptation of multimedia resources the ENTHRONE project proposes so-called adaptation Television and Multimedia (TVM) processors controlled by EIMS subsystems via Web Services. Although dynamic adaptation is supported, ENTHRONE does not implement appropriate interfaces towards IMS (or NGN) apart from nearly the same acronym. Adaptation using TVM processor may be not scalable for large number of users connecting the network for diverse services.

The OPTIMIX project studies innovative solutions enabling enhanced video streaming for point to multi-point in an IP based wireless heterogeneous system, based on cross layer adaptation of the whole transmission chain. The aim of the project is to increase the perceived quality of service for the user utilising the efficient cross-layer mechanisms enabling efficient joint approach between application and transmission.

The P2P-Next integrated project will build a next generation Peer-to-Peer (P2P) content delivery platform. The objective of P2P-Next is to move forward the technical enablers to facilitate new business scenarios for the complete value chain in the content domain from a linear unidirectional

push mode to a user centric, time and place independent platform paradigm. A platform approach allows modular development and modular applications, enables knowledge sharing and facilitates technology integration, code and skill re-use. This translates to fast development of new content delivery applications that build value for service and content providers. Regarding the content adaptation aspects in P2P-Next, the project aims to adapt the content at provider's level as obtained from a content source prior to the distribution in the system. The adaptation in P2P-Next can be related to the content format, packaging, linking to or combining with other content, etc.

European Union FP7 Integrated Project "MediA Ecosystem Deployment Through Ubiquitous Content-Aware Network Environments" (ALICANTE) proposes a novel concept towards the deployment of a new networked Media Ecosystem. The proposed solution is based on a flexible cooperation between providers, operators and end-users. Content-Awareness, Network-Awareness and User Context Awareness constitute three major challenges of the ALICANTE project, which imply necessary adaptations on the media services at different locations of the media service delivery chain during their delivery time. When the SP/CP Server starts to deliver the requested media service, it first makes the necessary arrangement in order to ensure that the media service is delivered with the necessary media service components in conformance as much as possible with the User preferences, for instance the language used by the media service. This kind of adaptations is called as "Media Service Personalisation", which do not imply any quality modification of the media services. Then, according to the Terminal capabilities (for instance, the screen resolution) and the Network conditions (for instance, the actual available bandwidth), the system may apply the necessary adaptations to the media service in order to ensure the best Quality of Experience (QoE) to the End User. This kind of adaptations is called as "Media Service Adaptation", which imply the quality modification of the media services.

The COAST (Content Aware Searching, Retrieval and Streaming) project aims to build a Future Content-Centric Network (FCCN) overlay architecture able to intelligently and efficiently link billions of content sources to billions of content consumers and offer fast content-aware retrieval, delivery and streaming, while meeting network-wide Service Level Agreements (SLAs) in content and service consumption. This will be achieved by combining intelligent network caching, searching and network, terminal and user context awareness. The COAST framework not only identifies what are the "*best*" host/cache and the end-to-end path but also find the content that best fits the user query. This may also include issues like adaptation and context (terminal, network, location) awareness. The content may need to be adapted before delivered to the user or even interactively or dynamically adapted. The reason for that may be the user preferences or the context (e.g. the user terminal, network, location, time of the day etc.).

OCEAN (Open Content Aware Networks) is another EU project aims to design a new open content delivery framework that optimises the overall quality of experience to end-users by caching content closer to the user than traditional CDNs do and by deploying network-controlled, scalable and adaptive content delivery techniques.

The SARACEN (Socially Aware Collaborative Scalable Coding Media Distribution) project aims to research and develop a platform, over which distribution of multimedia streams can be supported through innovative techniques, both as regards media encoding, but also in regards to media distribution. To achieve this, the project will make use of scalable media coding techniques including both standard and state of the art research methods (wavelets, multiple description coding), combined with new transport and real time streaming protocols deployed over peer to peer architectures.

4.2.4 Conclusion

These abovementioned projects intended to support different adaptation functionalities. However they are missing the opportunities to facilitate different emerging multimedia applications over large scale networks. In ENVISION, we propose to support adaptive multimedia services to heterogeneous clients using cooperation between the underling network and the overlay network. By relaying on ENVISION interface (CINA) the application is aware of different information including network topology, network condition. These overlay networks are constituted of different heterogeneous clients interconnected with each other through different heterogeneous networks. The CINA Interface helps client to be organised by taking into consideration media awareness, topology awareness, locality-awareness, QoS awareness, etc. it enables to determine means for dynamic adaptation according to the network's and terminal's capabilities more intelligently because it has explicit knowledge of the different parameters affecting the perceived content delivery quality. The dynamic content adaptation will be applied after extracting information at different level (Cross-layer information) and facilitated by locating the best adaptation gateway to provide the improved QoS to end-users. The overlay-based organisation and eventually content adaptation provides means for better scalability and QoS.

4.3 Requirements for Content Adaptation

Content Adaptation in ENVISION is subject to the following requirements:

- Content adaptation could be performed at the source, gateway and/or the destination nodes.
- The application generating the content should be able to adapt the content before transmitting. If that is not possible, the ENVISION application should select a particular adaptation node (adaptation gateway) to perform this task.
- The adaptation node should be able to perform the adaptation at two epochs. It can be performed at service invocation phase and/or at service delivery phase.
- The adaptation node should have the capability to ingest content with the supported formats (MPEG-2, H.264 and SVC) and with multiple data rates (bitrates) in a single layer coding, a scalable layer coding, or multiple description coding.
- The delivered service quality to the end user should be according to the user context (profile information such as user preferences, terminal capabilities, network condition, etc.)
- The adaptation of encrypted content requires the access to the encryption key. For such content, the encryption key should be provided to the adaptation node.
- Content adaptation must interface with the overlay content distribution functions and receive information about the active content consumers and dynamic updates about the overlay performance, including:
    - Throughput statistics for a set of nodes that receive a particular content object.
    - The number of content consumers for a particular content format, bitrate or SVC layer.
    - Arrival and departure of content consumers that do not support a particular protocol or encoding format and require the invocation of content adaptation functions.

    Based upon this information, content adaptation will be able to decide whether, where and when to trigger an update of the encoding profile, the addition or dropping of SVC layers, or the activation of a new transcoding session at the content adaptation node.

4.4 ENVISION Content Adaptation Specification

Content and service provisioning become problematic as many of the access devices are restricted in terms of processing, information display and network data rates, therefore, content must be suited to the user's preferences, usage context, device capabilities and network conditions. Since content may have to be converted into several intermediate forms to get into its final desired adapted form, content adaptation in ENVISION becomes multi-step process involving a number of services each performing a specific adaptation operation. The ENVISION content adaptation process mainly comprised of two functionalities. These are Adaptation Execution Function (AEF) and Adaptation Decision Function (ADF) as described in Figure 21. The input metadata is used to steer the adaptation process and the output metadata is used to update the information related to the adapted content. The following sections describe these two functions.

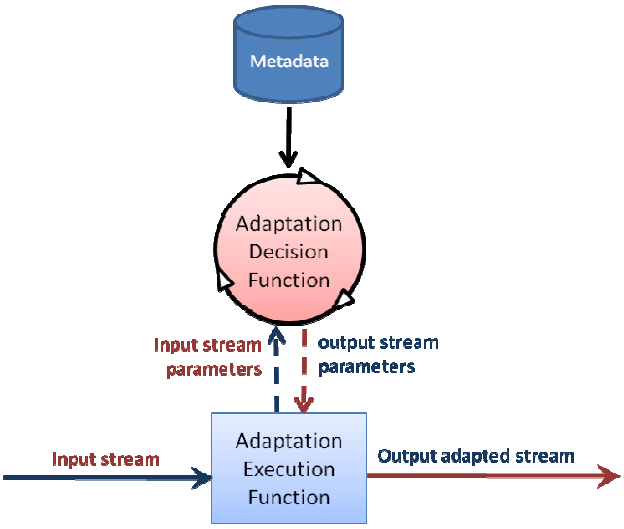


Figure 21: High Level Architecture of ENVISION Content Adaptation Process

4.4.1 Adaptation Execution Function (AEF)

The adaptation engine takes as inputs different content formats (MPEG-2, H.264, SVC, etc.) having different characteristics (for example bitrate and resolution) at different levels of bitrate and process them according to the desired parameters. In order to achieve this task, the adaptation engine starts first by acquiring the different necessary metadata. On the basis of these metadata the Adaptation Decision Function provides appropriate parameters for the output stream. Finally the AEF will perform the adaptation using these parameters.

4.4.2 Adaptation Decision Function (ADF)

The adaptation decision-taking in ENVISION is referred as the process of finding the optimal parameter settings for multimedia content adaptation execution given the properties, characteristics, and capabilities of the content and the context in which it will be processed. The Adaptation Decision Function (ADF) is a component that is able to take adaptation decisions based on available metadata. Adaptation decisions are mainly a set of values that are used as parameters for steering the encoding or the adaptation mechanisms within the adaptation node (i.e. service node). The metadata describes all different profiles and capabilities which are required to determine the optimum adaptation parameters. Hence, the computation within the ADF can remain generic and can be achieved by extracting the optimisation problem from the metadata and solving it. To make decision, several methods can be considered. (1) Adaptation decision function using utility function which assigns a weight to each input parameter (packet loss may have more weight

compared to end-to-end delay) in order to satisfy requirements expressed by different metadata. (2) Fuzzy logic introduced by [Z65] remains another alternative. This approach aims to reduce the constraints by combining the initial adaptation parameters ("*fuzzification*" process) and then take decision based on these new parameters. (3) Learning-based approaches can be considered also in order to take the adaptation decision. Artificial neural networks [RTV06] are an example of these methods. (4) The optimisation based model views ADF as an optimal problem with several constraints. The main challenges of the model are to devise proper target function and the resolving algorithm [MDK04]. These four widely used methods are under consideration for ADF in ENVISION.

Two approaches to perform the adaptation decision can be considered in ENVISION: centralised ADF or distributed ADF.

4.4.2.1 Centralised ADF

In centralised ADF, the adaptation decision making is performed by a single dedicated node. This node has access to all metadata elements essential for performing decision (Figure 22). This approach provides an optimal decision for adaptation. However, the computational cost in this case is on the higher side.

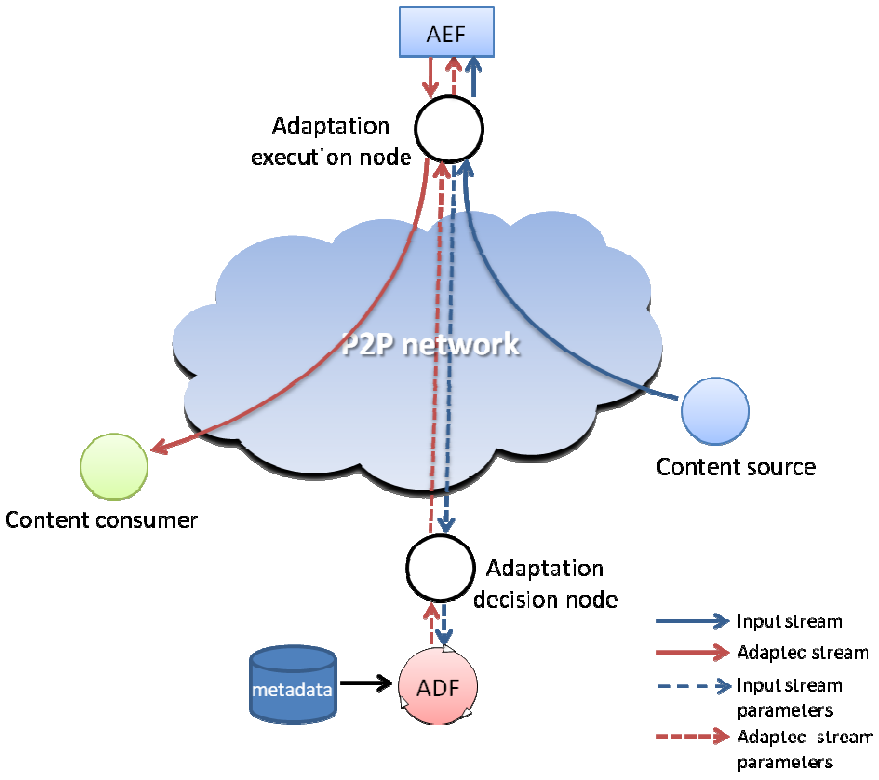


Figure 22: Centralised Adaptation Architecture

4.4.2.2 Distributed ADF

The distributed ADF provides an alternative that partially distribute the decision making among several nodes (Figure 23), in order to reduce the associated computational cost on the adaptation node. Each node involved in the adaptation decision making process has partial access to metadata parameters and determines partial output decision parameters. The distributed ADF remains more reliable due to distribution of tasks among several nodes, but less efficient since each ADF has not access to complete adaptation parameters.

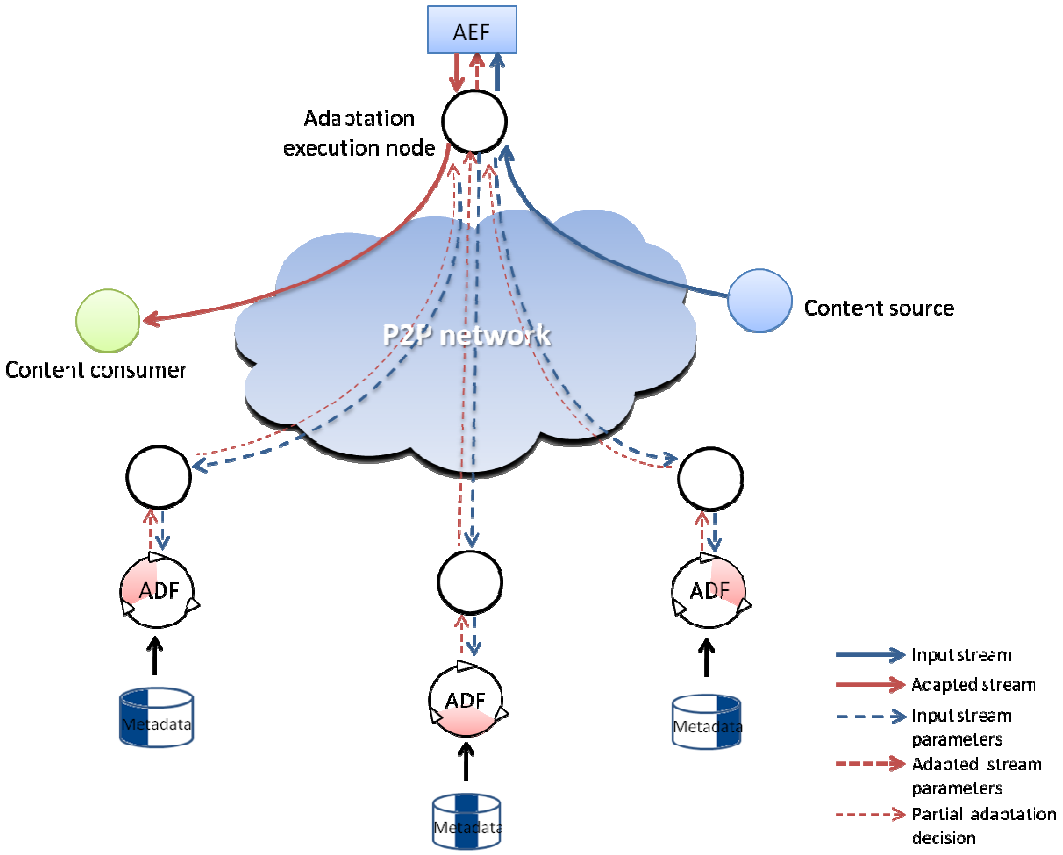


Figure 23: Distributed Adaptation Architecture

Formally, the two architectures for adaptation: centralised and distributed, can be compared based on the following criteria:

- Adaptation waiting time: time elapsed between detection of the adaptation event and the achievement of the task.
- Signalling overhead: represents the messages exchanged between adaptation execution node and adaptation decision node(s) in order to perform the adaptation.
- Distribution of overhead over nodes.
- Scalability of the architecture: it is the capacity to scale, i.e., the capacity to accept new users.
- In centralised architecture the adaptation decision is performed in a single Adaptation Decision Node (ADN), which can be different from the Adaptation Execution Node (AEN). In this case, the input stream parameters should be sent to the ADN in the network. This allows discharging the AEN from the decision computational cost. However, it causes signalling overhead due to messages exchanged between the ADN and AEN (Figure 22). In centralised architecture, it is difficult to scale to a large number of users since the ADN has to manage the whole metadata, to keep it update, and to perform all the decision computation operations. Another weak point of this approach is that the ADN represents a single point of failure.
- Alternatively, the distribution of the computational cost over different ADNs remains the major advantage of the distributed architecture. Consequently, the adaptation waiting time is reduced depending on the number of involved ADNs. Moreover, the distribution of metadata over different nodes mitigates the problem of metadata storage and management. However, the signalling overhead is greater than centralised approach since several ADNs, distributed over the network, are involved in the decision process. The decision needs to be performed in

coordinated manner to generate a harmonised output. The distributed ADF reduces the accuracy of the adaptation decision making, since each node has only access to a part of adaptation parameters.

- Both architectures, centralised and distributed, will be investigated in more detail in the context of ENVISION.

4.4.3 Where to Adapt the Content?

Another important question raised is where to adapt the content in case of distributed and heterogeneous overlay organisation in which different peers act as client consuming the content or servers providing the content. Some of them may also be just simple nodes relaying the content to other. In this case, the adaptation in ENVISION can be performed at different levels: at the content source, at the content consumer or at a relaying node level. In this sub section we will present the different alternatives with their pros and cons. The mechanisms and strategies to select the appropriate entity to perform the adaptation is the scope of the ENVISION deliverable D4.1 [D4.1].

4.4.3.1 Adaptation at Original Content-Source Level

In this approach, the adaptation is performed at the content provider (Figure 24) which has significant knowledge about network and terminal capabilities in order to decide and execute efficient adaptation. This approach provides more control over the multimedia content by limiting the alteration of the content owner. However, this approach is not scalable because the content provider takes into consideration of both the content provision and content adaptation. In addition, the adaptation of content is resources consuming (CPU, memory, etc.) which limits the number of consumers that can be supported by content provider. Also, the original quality (without adaptation) cannot be delivered as it to some capable receivers if an adaptation is applied.

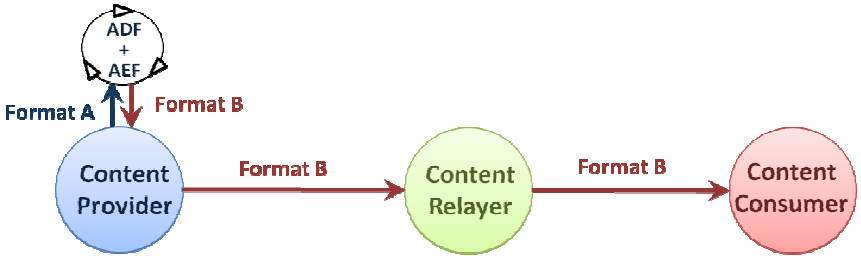


Figure 24: Adaptation at Original Content-Source Level

4.4.3.2 Adaptation at Consumer Level

In this approach, the adaptation is carried out by the content consumers (Figure 25) on the basis of some capabilities. The advantage of this approach is that the capabilities of the terminal are well known to the adaptation engine. However, adaptation does not take into account the capability of the network. Moreover, tiny portable terminals that have limited capability in terms of resources (computation power, memory) cannot perform adaptation.

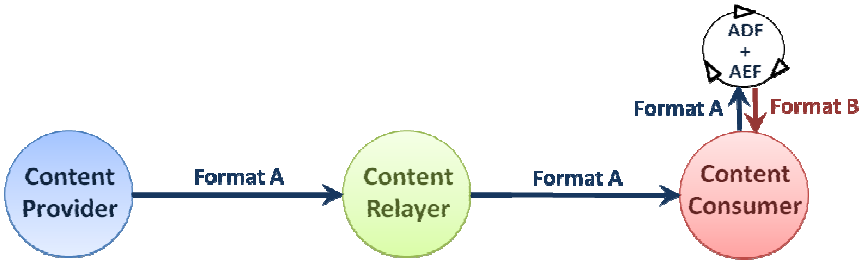


Figure 25: Adaptation at Consumer Level

4.4.3.3 Adaptation at Gateway Level or Intermediate Node

In this approach, the adaptation is performed at a dedicated gateway inside the network that is located between the content server and the consumer (Figure 26). This approach reduces the adaptation burden on content server and facilitates the content adaptation following the network characteristics. The deployment of adaptive gateways improves the scalability because each gateway performs adaptation for certain number of content consumers. However, this approach adds significant overhead while performing adaptation of secure media content because it requires decrypting the content before adaptation and then the resulting adapted content should be encrypted again to be delivered to the consumer.

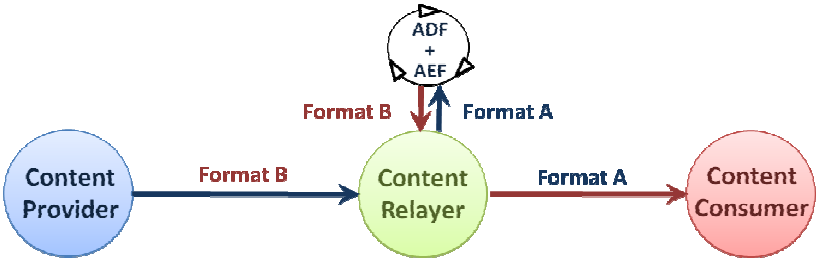


Figure 26: Adaptation at Gateway Level or Intermediate Node

4.4.4 When to Adapt the Content?

The epochs of service are also another important aspect to consider in the context of ENVISION. This is about the timescale related to different stage of the service, i.e. service invocation and service delivery/consumption. The adaptation scale should be accurate to reflect any change in the user context such as degradation of network conditions, changing in usage environment, etc. Adapting the content too often may not be a good solution for providing a good quality of experience as the quality may vary considerably over the time. Whereas, adapting the content irregularly (for example too late after detecting the problem) may affect considerably the received quality. So timescale has to be carefully taken into account to efficiently deliver a smooth, stable over the time, and acceptable level of quality of service/experience.

4.4.4.1 At Service Invocation

During the standard service subscription phase, the user specifies a set of metadata, which is mapped into a requested quality of service. At this point of time, the content parameters and adaptation parameters are used to retrieve an expected output from the decision taking function. The task of the adaptation-decision function is to find parameters for both the application (service) level and the network level that maximise the quality of service value under the given constraints. During the service invocation, the user may have different set of metadata and then the service needs to be adapted to this new user context. This operation is very important since it specifies the initial parameters for retrieving the service. It is considered as long term adaptation since it depends on initial context of the user. Starting from this point, the service will be delivered to the end user and should be adapted accordingly.

4.4.4.2 At Service Delivery/Consumption

The adaptation decision function during the service delivery phase is characterised by the fact that the metadata including network level parameters are dynamic and change frequently. The different metadata parameters are bounded by limitation constraints. Again, the goal of the adaptation decision function is to find parameters at the application-level and network level that maximise the quality of service value within the bounds determined by the selected quality. This adaptation is considered as short term in which, after detection of service perturbation (e.g. changing in network

conditions), analyses the situation and determines a temporal adaptation strategy. It can, for example, temporary reduce the number of stream layers previously decided by the long term adaptation algorithm, in order to adapt to the new changing context. Then, the short term adaptation tries to return back to the ideal situation fixed initially or the one that can be consumed.

Figure 27 shows the changes in quality levels due to the varying user context including network conditions and metadata information related to the user. The user specifies the desired quality of service during the service invocation phase. During the reception of the service, a certain change in network conditions (e.g. decrease in bandwidth) affects the desired quality level of the user. The adaptation at service delivery/consumption allows coping with this type of dynamic network changes and ensure the smooth and acceptable delivery of service.

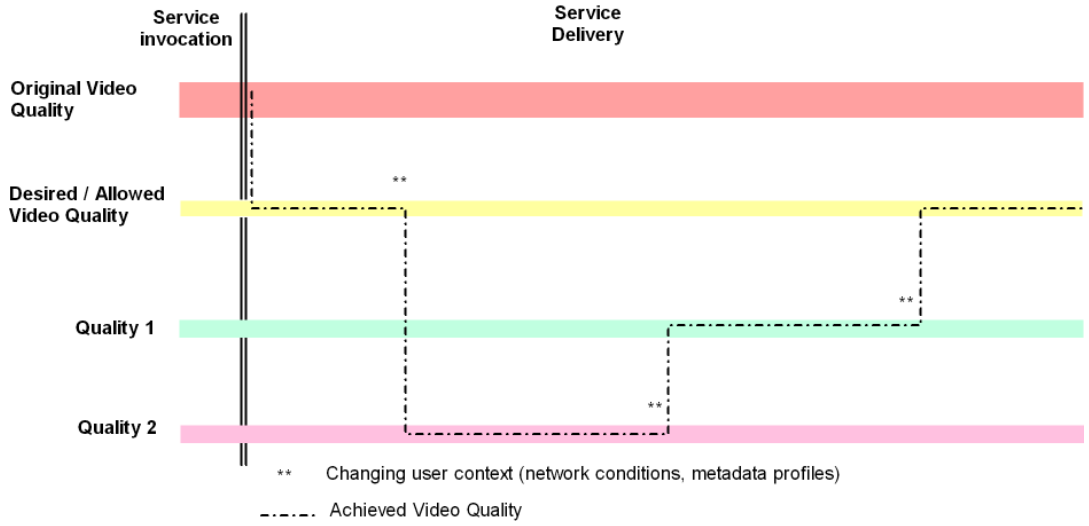


Figure 27: Achieved Quality Levels during Service Invocation and Delivery

4.4.5 How to Adapt the Content?

The last question related to content adaptation concerns the techniques that may be used for content adaptation. Content delivery in ENVISION needs to address both the multimedia nature of the content and the capabilities of the diverse client consuming the content. The Adaptation Execution Function (AEF) is able to adapt the content format/bitrate according to the parameters computed by the Adaptation Decision Function (ADF). This can be achieved using transcoding, transrating, and/or SNR adaptation techniques. However, emerging of new AV encoding standards as SVC (3.2.4), has opened up new horizons for adaptation which consist of simply dropping of enhancement layer. Yet which layers to drop is considered as a challenging problem and need to be investigated further as many possibilities exist when layer are dropped.

4.4.5.1 Codec Adaptation (Transcoding)

This adaptation function in ENVISION will allow the transcoding from a certain format to another one. For example, an original MPEG-2 content can be transcoded to SVC to enable scalable content distribution for heterogeneous consumers with different profiles. Similarly, SVC content can be transcoded to an MPEG-2 in order to serve a legacy terminal which accepts only MPEG-2 format. Mainly we can distinguish the following configuration as we are only concerned by MPEG-2, H264 and SVC codec but this can be easily extended to other AV formats (see Figure 28):

- Adaptation to MPEG-2 (legacy terminal): for example from H.264 to MPEG-2 or from SVC to MPEG-2
- Adaptation to H.264: for example from MPEG-2 to H.264 or from SVC to H.264

- Adaptation to SVC: for example from MPEG-2 to SVC or from H.264 to SVC

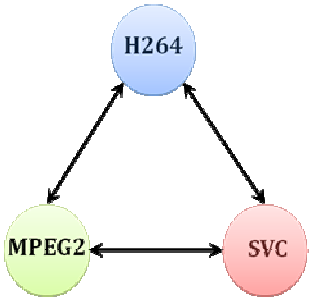


Figure 28: Example of Codec Adaptation

4.4.5.2 Bitrate Adaptation

The bitrate adaptation in video streaming context is the modification of the stream bitrate in order to meet the network bandwidth requirements or the receiver device capability/preferences, while keeping a QoS/QoE above a certain threshold. This adaptation can be performed by reducing or enhancing the fine-grained video quality (Figure 30). It is also known as the SNR (Signal to Noise Ratio) adaptation. The bitrate adaptation can also be performed by reducing/increasing the spatial resolution of the video, i.e. *spatial adaptation* (Figure 33). Finally, the bitrate adaptation can be performed by deleting some frames from the video (Figure 34), and consequently reducing the framerate, defined as the number of frames played per time unit. This is called *temporal adaptation*. In the majority of the cases, SNR adaptation is more suitable since it allow to easily targeting the desired bitrate while maintaining an acceptable level of user experience. Also, reducing frame rate of some video content (football match, action movies, etc.) may affect considerably the user experience. In this case, it is preferable to have a lower SNR than lower frame rate.

4.4.5.2.1 Quality (SNR) Adaptation

The quality adaptation (Figure 30) or Signal to Noise Ratio adaptation process relies on the principle of image compression techniques.

The best representation of an image in computer is the RGB format (Red, Green and Blue). With this format, each pixel constituting the image is coded into three colour components (8 bits per colour component). Since this representation is memory consuming and considering the fact that the human vision system is much more sensitive to brightness than to chrominance, the image is converted from RGB into a different colour space YCbCr (YUV). This format has three components: Y is for the brightness (luma) of a pixel, while Cb and Cr represent the chrominance of the pixel (Blue and Red, Green is then deduced from the two chrominance components). This format allows a great compression without an effect on perceptual image quality. If an instance format of YCbCr is 4:2:2, then it specifies that for 4 pixels, the Y component is coded into full resolution (i.e. 32bits/8 bits per pixel) and the other two components are coded into a down-sampled format (i.e. 16 bits for CbCr/8 bits per component, shared between two pixels). The next step in the encoding process is the splitting of the image into square regions (blocks) of 8 pixels per side (4x4 or 16x16). Each block is converted to a frequency domain representation using a normalised, two-dimensional type-II discrete cosine transform (DCT). After computing the DCT matrix for a block, the quantisation step occurs. This step aims to reduce the overall size of the DCT coefficients. At this stage, small parts of the data will be lost. This step consists of dividing DCT coefficients by values stored in an 8x8 quantisation matrix and then rounded to the nearest integers. The value of the quantum is inversely proportional to the contribution of the DCT coefficient to the image quality. After the quantisation step, the obtained matrix will be scanned (Scanning step) in order to be efficiently compressed (Compression/Encoding step) using an entropy coding algorithm (VLC, RLE, Huffman, etc.). The high level encoder block diagram is shown in Figure 29.

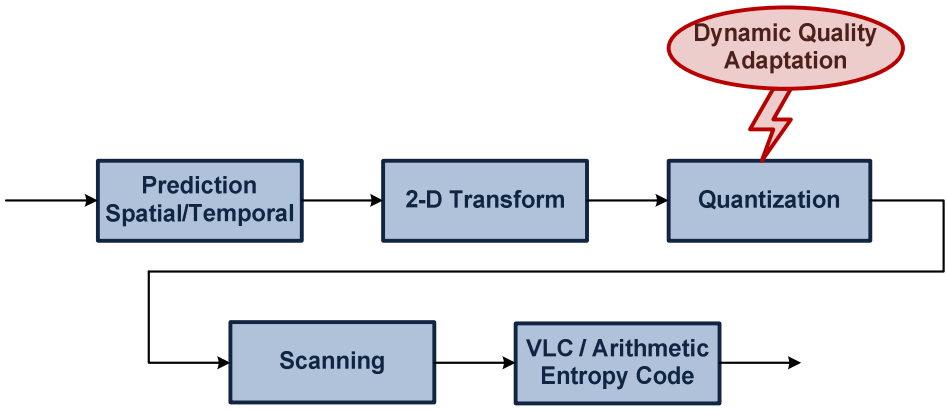


Figure 29: Example of High-Level Encoder Block Diagram with Quality Adaptation

The idea behind bitrate adaptation based on quality adaptation is to act dynamically on the quantisation step (Figure 29). In fact, quantisation parameters can be adapted dynamically in order to meet the bitrate requirements. However, as the dynamic quantisation adaptation occurs at the image level, it has to be done smoothly according to the previous images in order to limit the variation (level of blurring). Thus, an equation that takes into account the variation regarding the previous images and the targeted bitrate has to be resolved to obtain the best quantisation parameters. An example of quality adaptation is given in Figure 30. Figure 31 and Figure 32 represent results of measurements of Peak Signal to Noise Ratio (PSNR) and Structural SIMilarity (SSIM) [WBSS04] for this example.

100% Quality **50% Quality** **10% Quality**

Figure 30: Quality (SNR) Adaptation

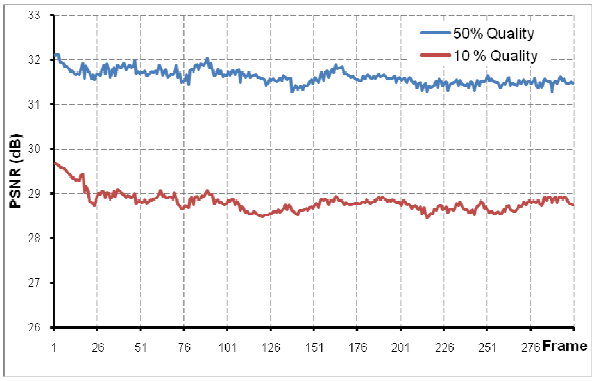


Figure 31: PSNR Measurement

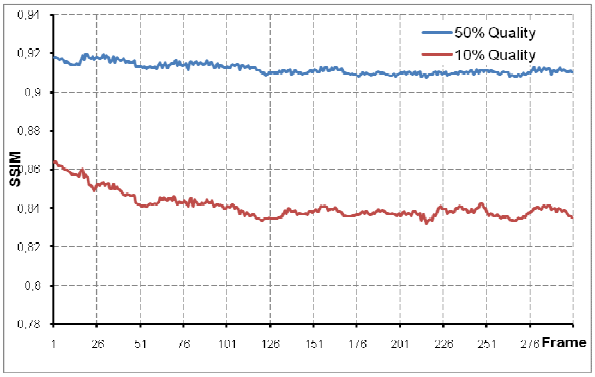


Figure 32: SSIM Measurement

4.4.5.2.2 Spatial Adaptation

The spatial adaptation (Figure 33) is based on image scaling (upscaling/downscaling). Some of the well known image downscaling techniques are discussed below:

- Nearest Neighbour Interpolation: Nearest-neighbour interpolation is a simple method of multivariate interpolation in one or more dimensions. Interpolation is the problem of approximating the value for a non-given point in some space, when given some values of points *around* that point. The nearest neighbour algorithm simply selects the value of the nearest point, and does not consider the values of other neighbouring points at all, yielding a piecewise-constant interpolant. Nearest neighbour requires the least processing time of all the interpolation algorithms because it only considers one pixel (the closest one to the interpolated point). The algorithm is very simple to implement, and is commonly used (usually along with mipmapping) in real-time 3D rendering to select colour values for a textured surface.
- Bilinear Interpolation: An interpolation technique that reduces the visual distortion caused by the fractional zoom calculation is the bilinear interpolation algorithm, where the fractional part of the pixel address is used to compute a weighted average of pixel brightness values over a small neighbourhood of pixels in the source image. Bilinear interpolation considers the closest 2x2 neighbourhood of known pixel values surrounding the unknown pixel. It then takes a weighted average of these 4 pixels to arrive at its final interpolated value. This results in much smoother looking images than nearest neighbour. Bilinear interpolation produces pseudo resolution that gives a more aesthetically pleasing result, although this result is again not appropriate for measurement purposes.
- BiCubic Interpolation: Bicubic goes one step beyond bilinear by considering the closest 4x4 neighbourhood of known pixels for a total of 16 pixels. Since these are at various distances from the unknown pixel, closer pixels are given a higher weighting in the calculation. Bicubic produces noticeably sharper images than the previous two methods, and is perhaps the ideal combination of processing time and output quality. For this reason it is a standard in many image editing programs (including Adobe Photoshop), printer drivers and in-camera interpolation.

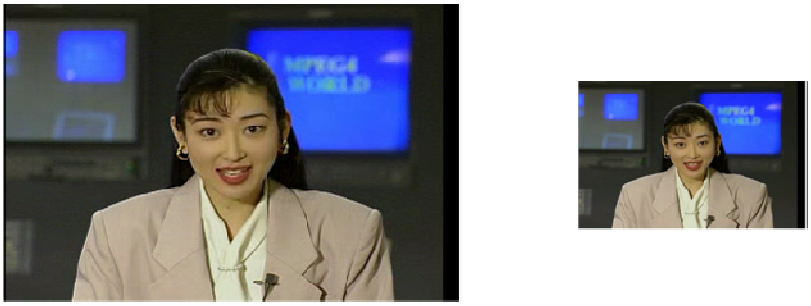
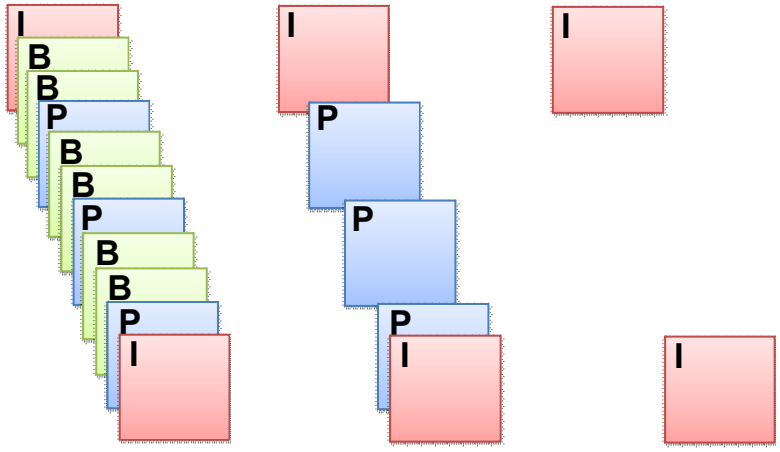


Figure 33: Spatial Adaptation

4.4.5.2.3 Temporal Adaptation

The temporal adaptation (Figure 34) of an AV content is performed by dropping some intermediate frames. The selection of frames to be dropped should be achieved taking in consideration the frame importance in the Group of Picture (GOP). Indeed B-frames are dropped first, since they are not referred by other frames (Figure 35.b). Then in second time, If desired bitrate is not exceeded, P-frames are dropped, too (Figure 35.c).

Figure 34: Temporal Adaptation



(a) Original MPEG-2 GOP structure before temporal adaptation

(b) MPEG-2 GOP structure after dropping B-frames

(c) MPEG-2 GOP structure after dropping B-frames and

Figure 35: Example of Temporal Adaptation in MPEG-2

4.4.5.3 Protocol Adaptation

Different content may be transmitted via different protocols. The adaptation service in ENVISION will include a protocol adaptation which allow clients that do not talk a particular protocol to access the service. For example, an end user behind a firewall would not be able to receive a content using RTP with RTSP signalling as the necessary ports may be blocked by the firewall. Thus, an intermediate node (i.e. service node performing protocol adaptation) involved in a distribution of a content (that is pushed using RTP by the content provider) may be switched to HTTP delivery in order to serve this user which is behind a firewall. A signalling mechanism is very important as the service may be delivered in a non-transparent manner.

Mainly we can distinguish but other protocols may be easily integrated (see Figure 36):

- Adaptation to MPEG-2 TS (transport stream): for example from RTP to MPEG-2 TS
- Adaptation to RTP: for example from MPEG-2 TS to RTP
- Adaptation to HTTP: from RTP to HTTP or from MPEG-2 to HTTP, for example: Apple solution (MPEG-2 TS encapsulated over HTTP)

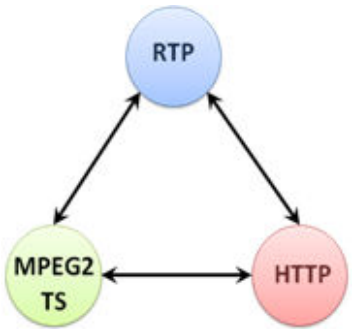


Figure 36: Example of Protocol Adaptation

5. ERROR RESILIENT AV TRANSMISSION

5.1 Introduction

A video communications system typically involves five steps. First, the video is compressed by a video encoder to reduce the data rate. The compressed bit stream is then segmented into fixed or variable length packets, which might then be transferred directly, if network links can be considered error-free). Otherwise, they usually undergo a channel encoding stage, typically using forward error correction (FEC). At the receiver end, the received packets are decoded, unpacked, and the resulting stream is then processed by the video decoder to reconstruct the uncompressed video. The overview of such a general FEC mechanism is shown in Figure 37.

Unless there is a dedicated link that can provide guaranteed delivery between the media source and destination, data packets may be lost due to traffic congestion or corrupted by physical bit transmission errors, as is common with wireless channels. One way in which error-free delivery of data packets can be achieved is through the use of ARQ (Automatic Repeat Request) protocols that retransmit lost or damaged packets**.** Such retransmissions, however, may introduce delays that are unacceptable for delay-sensitive applications such as interactive live video conferencing. In these cases, excessively delayed packets are considered lost. Further, many broadcast and multicast applications prevent the use of retransmission algorithms due to network flooding considerations.

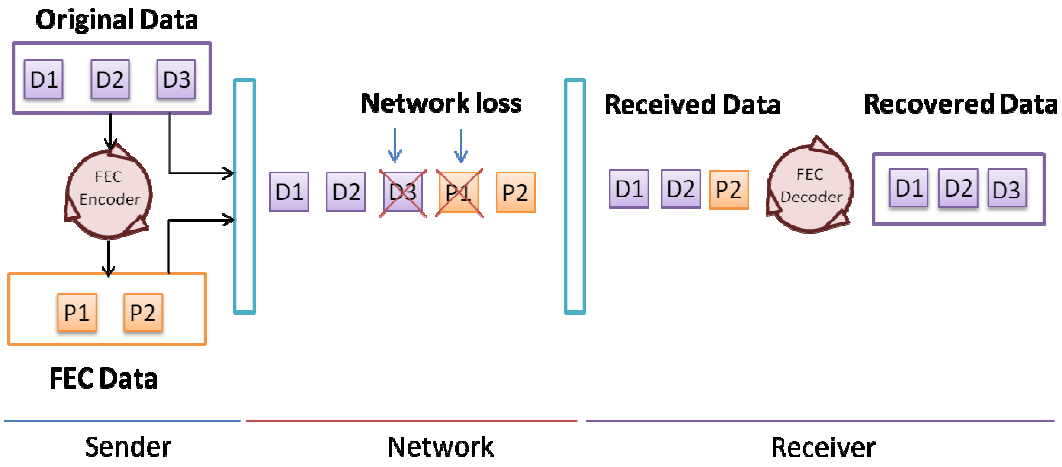


Figure 37: Overview of FEC Mechanism

In this section, we present error resilience coding mechanisms for packet video delivery. This is an important building block for ENVISION applications, as it helps P2P delivery to guarantee an acceptable level of QoS as the network scales to a large number of users (a taxonomy of reliable video delivery techniques is shown in Figure 38). We start this next section by presenting a state of the art related to error-resilient AV transmission; we then present an initial specification of ENVISION error resilience which benefits from cooperation between applications and the network through the use of the CINA interface.

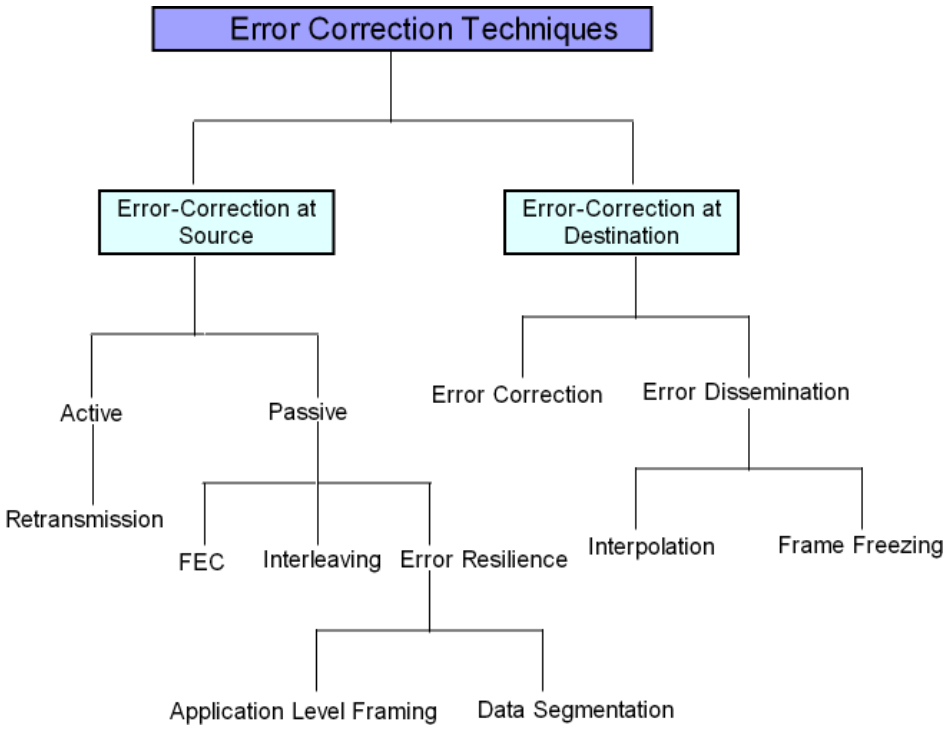


Figure 38: Taxonomy on Reliable Video Delivery

5.2 State of the Art

5.2.1 FEC at the Transport Layer (L4)

The use of error correction in the transport layer has several advantages over other approaches. First, it allows the inclusion of error correction without rewriting applications. In addition, it requires only the end stations to handle error correction code, adding no additional packet processing overheads at routers or switches along the delivery path. Finally, by performing option negotiation at connection setup time (and by virtue of not imposing changes to intervening network elements), it can be gradually incorporated into the network. This transparency of transport layer FEC simplifies coding design and provided implementation flexibility by allowing both software and hardware solutions.

Although the transport layer is a natural candidate for end-to-end error correction, FEC has also been used at the physical and the data link layers to provide single hop reliability. This opens the question of whether FEC should be implemented at the transport layer, or delegated to hop-by-hop implementations. This last scenario will lead to less efficient implementations, particularly when bottleneck link congestion avoidance is considered. If we assume that TCP would undergo no special changes, the deployment of FEC options could be particularly beneficial by eliminating slow-start episodes triggered by correlated packet loss. Thus, when considered in addition to usual retransmission-based error control strategies, layer 4 FEC can provide increased reliability and throughput.

5.2.1.1 Placement of the FEC Sublayer

Multimedia data transmission on the Internet often suffers from delay, jitter, and data loss. Data loss in particular can be extremely high on the Internet, particularly when it accompanies link failures or excessive congestion events. The approach of TCP is to ensure the error-free delivery of all datagrams. However, unlike traditional applications, multimedia applications can tolerate some data

loss: small gaps in the media stream may not significantly impair media quality. Thus, alternative error control methods optimised for media streams have been proposed as substitutes to TCP.

There are two obvious possibilities for the incorporation of FEC schemes into the transport layer: either by coding layer 4 datagrams in their entirety, or by only coding datagram payload [ACD04]. In the first case, the transport layer output is sent to a FEC sublayer before IP layer. The sender's FEC sublayer then expands the output by adding M parity packets for each N data packets, and submits the expanded data stream for IP forwarding. The receiver FEC sublayer of the receiver then receives the packets and extracts the original data if sufficient packets have been received. The reconstructed data is delivered to the transport layer, which is oblivious to any missing packets that had been corrected. If the original data cannot be reconstructed, the receiver responds as it would to a regular packet loss event (e.g. recover the encoded lost packet by retransmission).

In the second case, data sent by the application is expanded by the FEC sublayer to incorporate additional error correcting packets. The data is then numbered and submitted to the transport layer. On the receiving side, the transport layer needs to be given additional information to decide whether a loss event is severe enough that retransmissions need to be triggered, or whether it should simply forward the incomplete data stream to the FEC sublayer which then performs error recovery. Normally, this is done by FEC logic that identifies whether enough datagrams have arrived (in which case they are forwarded to the transport layer with an indication to ignore any potentially missing packets that may trigger retransmission). If the number of datagrams is insufficient for FEC recovery, the transport layer will handle any retransmissions.

The use of FEC at the transport layer allows error control without the need for the additional delay and jitter produced by retransmissions and the window-based sending of data, which typically make TCP typically unsuitable for interactive multimedia applications. For these cases, the "best-effort" service provided by UDP gives the multimedia application greater control over timing and can be complemented with FEC to provide data loss guarantees [FC01]. The literature on FEC with UDP can be analysed in terms of the degree of knowledge that the FEC logic has on the data that is being transported. *Media independent* FEC seeks to repair data as a binary stream, with no knowledge of the data being transmitted. These systems have some shortcomings when they are used for interactive sessions, the primary of which is their added end-to-end delay. To see the reasons for this, consider a loss episode: the receiver will need to wait until it has received a number of packets to be able to reconstruct any missing data. This will increase the overall playback latency in loss-free conditions, because the receiver will need to receive many packets before being able to decode the data stream. *Media-specific* FEC uses knowledge of the data when computing encoding information. Low bandwidth redundancy channels are included within the video stream at the sender, and information to recover from packet losses is selectively inserted in these channels when needed. If these channels are insufficient to recover from especially severe loss episodes, a repetition-based error concealment technique is used to fill the gap in the media stream. Their low delay and high reliability makes media-specific FEC techniques a good choice for interactive applications where a large end-to-end delay is a concern.

5.2.1.2 FEC Code Allocation

One of the methods to increase the robustness of video transmission is by adding FEC codes to the compressed video bitstream [CLK09]. However, adding FEC codes to the compressed video bitstream comes at a price of adding redundancy back to the bitstream, which is a conflict of reducing redundancy in the video during the compression process. In this case, the video source has to be further compressed in order to reduce its output bitrate further to accommodate the extra bitrate needed to add FEC codes to the compressed video bitstream. In general, there is a trade-off between redundancy and error robustness, both at bitstream transmission and media playback. The compressed bit stream of a video takes all the redundancy from the stream, mainly by utilising the

correlation between neighbouring frames in a differential manner. This increases the sensitivity of the stream prone to errors dramatically, while the errors propagate to the next frames.

One of the well-known FEC codes is the Reed-Solomon (RS) code [RS60]. RS is very well suited for error protection against packet loss, since it is a maximum distance separable code, which means that no other coding scheme can recover lost source data symbols from fewer received code symbols. The aim of Reed-Solomon (RS) codes is to produce at the sender *n* blocks of encoded data from *k* blocks of source data in such a way that any subset of *k* encoded blocks suffices at the receiver to reconstruct the source data.

In transporting real-time media over IP networks either UDP or RTP (Real Time Transport Protocol) protocols can be used. RTP provides packet sequence ordering over UDP, enabling a receiver to identify out of sequence, discarded or reordered packets, which makes it more robust than UDP. The Pro-MPEG [PMPEG11] FEC scheme uses the RTP transport protocol as a building block for providing packet recovery techniques to ensure reliable real-time media transport.

Normally, the generation of the FEC packets is based on the use of a matrix. The size of this matrix is defined by two parameters L and D, where L is the spacing between non-consecutive packets to be used to calculate the FEC packet and D is the depth of the matrix. The size of the matrix implements a given trade-off between latency, transmission overhead and error protection. Once the media packets are aligned in the matrix the FEC packets are computed by XOR (exclusive or) of the media packets along the column or row of the matrix. Column FEC provides correction for consecutive burst packet loss of up to L packets. The FEC packets are generated per a column within the matrix allowing loss of any single media packet within a column or burst of error within a row to be corrected through the FEC packet. Column FEC is ideal for correcting packet burst errors and random errors. Row FEC provides correction of non-consecutive packet loss and can correct any single packet loss within a row of media packets. The FEC packets are generated per a row allowing loss of any single packet to be recovered. Row FEC is ideal for correcting random packet errors.

5.2.1.3 Unequal Error Protection

It is a well-known fact that the binary bits in a compressed video bitstream are not equally important, with some of the binary bits having higher importance compared to other binary bits. For example, the video bitstream's header is much more important than the DCT coefficients data and thus should be better protected against transmission errors. Unequal Error Protection (UEP), which is based on priority encoding transmission, is a transport level error control technique in which the more important bits in the compressed video bitstream are better protected against transmission errors compared to the less important bits [BNZ08]. There are many approaches in which the more important video bitstream can be better protected against transmission errors compared to the other less important video bitstream. The two commonly used approaches are by using FEC codes and hierarchical modulation in order to have different protection orders to the compressed binary bitstream (hierarchical modulation is not considered in this section). For UEP using FEC code, a stronger FEC code is allocated to the more important sections of the compressed video bitstream than to the less important ones.

UEP can be mainly classified into three categories according to the consideration of different aspects of the compressed video bitstream's sensitivity to transmission errors. The three UEP categories are:

- UEP using different importance of binary bits in a video bitstream
- UEP using different importance of frames in a GOP (Group Of Pictures)
- UEP using different importance of layers in scalable video coding

5.2.1.4 Interleaving

Interleaving is a tool that can be used in digital communications systems to enhance the random error correcting capabilities of block codes such as Reed-Solomon codes to the point that they can be effective in a burst noise environment. The interleaving subsystem rearranges the encoded symbols over multiple code blocks. This effectively spreads out long burst noise sequences so they appear to the decoder as independent random symbol errors or shorter more manageable burst errors. The amount of error protection based on the length of the noise bursts determines the span length or depth of interleaving required [KTK10]. Interleaving can be classified as either periodic or pseudo-random. The periodic interleaver orders the data in a repeating sequence of bytes. Block interleaving is an example of periodic interleaving. These interleavers accept symbols in blocks and perform identical permutations over each block of data. One way this is accomplished involves taking the input data and writing the symbols by rows into a matrix with *i* rows and *n* columns and then reading the data out of the matrix by columns. This is referred to as a (*n,i*) block interleaver. Pseudo-random interleavers rearrange the data in a pseudo-random sequence. Periodic interleaving is more commonly invoked because it is more easily accomplished in hardware.

5.2.2 FEC at the application layer (L7)

Although the FEC at the transport sub-layer provides a good solution and may serve any upper layer (i.e. application), it is not common in today implementations. Alternatively, the use of FEC at the overlay layer provides a solution to any application instance that has the capability to join the overlay services. The state of the art provided in the above sections fits here as well, with minor changes. Another benefit from L7 FEC is that the application layer also has the best knowledge of application specific information, such as the content being delivered, its urgency and the required buffering states at the receiving end that will ensure smooth streaming. When P2P applications are considered for live video streams, the overlay may benefit from full control of the transmission including FEC at the application levels for the following reasons:

- Assuming the different source peers hold the same information/bit stream, they could collaborate together and avoid long interleaving schemes, taking benefit from uncorrelated transmission paths.
- Managing the delivered content parts and their protection levels is simpler and does not require lower levels collaboration with networking layers.
- FEC blocks delivered through several peers in a distributed manner may not provide all the information data to the network layers to perform the required FEC, making processing and data integrity more complex.

Some of these benefits are explained in the following section.

5.2.3 FEC in P2P for VoD Distribution

5.2.3.1 Multiple Descriptors - FEC

Multiple Descriptors – FEC (MDFEC) [PR01] is a popular coding scheme for multiple encoding of content, where multiple *descriptions* of the content are available. Different MDC video layers, for example, are candidates for MDFEC code. The aim of the code is to achieve unequal protection of different content and to allow some reconstructions of more important data in case of loss. In the I-Share project [I-SHARE], they used the MDFEC scheme to mitigate the phenomena that a peer becomes unavailable. MDFEC is based on RS codes where each content part is arranged as shown in Figure 39. As shown in the figure in part (a), 4 content parts/layers data are input for the encoder L1 to L4, in (b) the arrangement of the source information into two dimensional matrix, and in (c) the parity bits are generated. Generally in MDFEC codes, the numbers of rows and columns are M, and are equal to the number of existing layers. The rows are typically assigned to different peers (when

used for P2P delivery) in a way that each peer will deliver different and complementary code parts, providing new information to the decoder with no overlapping.

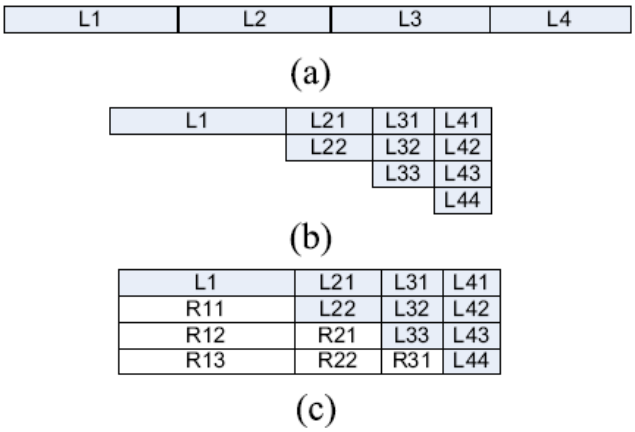


Figure 39: MDFEC Scheme

5.2.3.2 Redundancy-Free Multiple Description Coding and Transmission

Redundancy-Free Multiple Description Coding and Transmission (RFMD) [LSP07], is based on the MDFEC codes described above with some improvements. In many typical scenarios (like VoD) RFMD codes outperform other popular coding schemes such as MDFEC, single layer Reed Solomon (SLRS), and SVC. As shown in Figure 40, where the gray area is actually transmitted, (a) normal MDFEC as described above, (b) for the case that one peer is available, (c) for the case that two peers are available, (d) for the case that 3 peers are available and (e) for the case that 4 peers are available. As long as there are more peers available the quality of the video is increased. There is no dependency on which peer is dropped or become unavailable on the quality, unlike SVC stream where different peers deliver different layers of the code, in such case, there is a tight dependencies as higher layers cannot be displayed when lower layer is missing. Thus the RFMD scheme provides robustness to peers unavailability, allowing reconstruction of the swarm with moderate degradation of the quality.

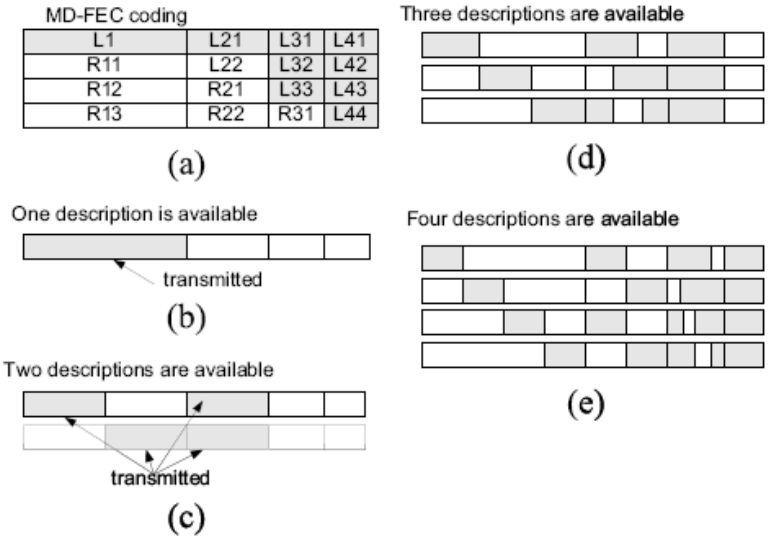


Figure 40: RFMD Code Illustration

5.3 ENVISION Requirements for Error Resilient AV Transmission

The following section describes some requirements for error resilient AV transmission. We will focus on Forward Error Correction (FEC).

- The FEC entity should generate, in real-time, code parities to match with link conditions of live interactive video services (i.e. loss probability, delay)
- The FEC should produce a level of protection to meet content priority; at least 3 levels of priority should be supported simultaneously.
- The FEC should be able to provide estimations of required overheads per link.
- The FEC should be able to provide extra parities to support additional available bandwidth.
- The FEC should generate code blocks optimised to reduce delay.
- The FEC should generate different parity codewords to support the P2P distribution of the same content from several end points simultaneously (boosting capacity, with reduced delay), this could be done using rateless codes (with no dependency between sources) or via other codes (with some dependency that should be addressed through the CINA interface)
- The FEC content should be cached and be used by multiple clients, with various channel conditions.

5.4 Initial FEC Specification for ENVISION

ENVISION FEC logic will be implemented to support the activities of live video delivery over P2P networks. We will further study whether to locate the FEC at the transport sublayer as discussed in section 5.2.1 or alternatively at the application layer as discussed in section 5.2.2. Moreover, the use of RMDF approach for live distribution with low latency will be studied. Attention will be paid to product code improvements as well as cross-interface codes for multilink-enabled devices. Unlike VoD, live stream is more sensitive to variations in quality, overlay connectivity and additional dynamic changes as both receiving and serving peers are actively involved at any given time (user behaviour will influence the availability of serving peers). Moreover, since delay sensitive applications cannot easily rely on retransmission mechanisms, we believe that FEC will have a greater role than in P2P VoD applications.

The use of the CINA interface to exchange FEC parameters will be explored, along with the possible place that the network might take in FEC encoding (this may be eventually reflected in the definition of CINA operations). In some cases the network could act as a peer and reconstruct the data, generate missing packets and so on, thus trading off network efficiency with complexity. The network, which in current ALTO implementations advices on a “cost” value for the communication between peers, could also take into account FEC attributes when computing the cost value. For instance, if some peers may not be FEC enabled (i.e. not ENVISION enabled peers), the network could recommend peers with FEC capabilities when network loss conditions are unfavourable. These attributes could be located at the network, as part of metadata it shall hold on the different peers.

We will also study the option of using FEC parity packets instead of dummy active monitoring packets for the purpose of link monitoring and grading. This approach could improve the performance of the network, as some monitoring jobs will be used as data and will save resources in comparison to the case where dummy payload packets are delivered. As mentioned above, the FEC is not only for ordinary packet loss cases, it will be involved in swarm creation (i.e. in the process where the decisions who serve what and to which peer are taken) and repair (in cases where conditions change) and thus would be reflected to the CINA interface. More specifically, as described in section 7, on boosting the uplink capacity, FEC will be used over multiple connections, thus we aim to study and make benefit of multilink enabled peers and develop code to meet their requirements.

6. CONTENT CACHING

6.1 Introduction

During the last years, Over The Top (OTT) application traffic (especially video) has grown significantly. This is in part due to the proliferation of Internet-connected devices (many with broadband network access capabilities, in excess of 10 Mbps), the increasing user shift towards Internet content and the availability of high quality long-form content. The result of these changes is that OTT video consumption is inducing considerable network cost and resource consumption. However, this OTT traffic is generating very limited income for the core and edge network operators. It may be argued that this is a problem to be solved by business models that provide adequate incentives for the improvement of network capacity. However, even if one takes this as granted, additional research topics must be considered, with caching being one of the most important ones. Historically, caching research has focused on static web content, but it has also addressed the management of large objects transported to enable bandwidth savings between network clients and remote content servers. In addition to these efficiency improvements, caching provides gains to end-user QoE. By placing the caches closer to the users, they can access cached content quickly and with low cost for the network provider.

The next subsection provides state of the art on caching mechanisms and how ENVISION can take advantage of them. We divided the caching mechanism literature according to the lifetime of the data inside the cache. This will lead to two type of caching mechanism: short and long term.

6.2 State of the Art

6.2.1 Short Term Caching

The basic idea behind the caching techniques is that if two clients request the same video at different instants, the source may serve the latter one using data that is already cached on behalf of the former one. Thus, the referenced video is read from the source only once, but can reach several simultaneous users. In other words, caching is a proactive policy which minimises the seek-to-transfer time ratio. We believe that an effective caching strategy must consider the following three factors:

- Coverage: The fraction of memory occupied by the cached content. The buffer should be distributed efficiently for caching video content.
- Accuracy: The cached content actually used by the peer. The term also referred as Hit Ratio and it shows the effectiveness of caching strategy.
- Timeliness: Data must be available to users before it is needed but not so early that it is discarded without being used.

Several techniques have been proposed for multimedia caching. Cheng et al [CJL07] proposed a scheme in which a scheduler adaptively fetches chunks to buffer periodically. The scheduler determines the number of chunks to be fetched after analysing the capacity of peers. Rejaie et al [RYH00] proposed a proxy caching mechanism for layered-encoded streams to maximise the delivered quality of popular streams among interested clients. Although caching of continuous media has been discussed, there is no concrete solution for caching content in random and unpredictable environment. Such environment is under consideration in ENVISION use cases described in bicycle race and web3D conferencing. An optimal offline caching algorithm and a heuristic caching algorithm were proposed in [SLP06]. It is shown that performance of layered video can be improved by applying appropriate caching policies and caching provides higher gain to the layered stream. In the next sub-sections, we will discuss the existing caching techniques in P2P streaming system.

6.2.1.1 Random Caching

The random caching [CJL07] is used to acquire the data in local cache before a seek operation is carried out. Rather than waiting for a cache miss to perform a request, random caching anticipates such misses and issues a request to local cache in advance. Every peer caches a few seconds' worth of recent data, which is replaced using a sliding window technique. If a peer wants to cache pre-fetched data, it will request the peers having a minimum *playhead* from it (the time difference of the current playing position between the two peers). Data chunks are requested randomly from other peers. The scheduler is responsible for caching segments in periodic intervals. If a segment is not available in the neighbourhood of the peer, it can be either requested from server or far neighbours. A cached segment, not consumed for a certain period of time will be discarded and a newly fetched segment is placed in buffer. Although caching of data is considered in random caching, it has not been addressed in unpredictable user behaviours. As a result more useless segments occupy the local cache. Furthermore, the unavailability of useful content increases access latency in random caching.

6.2.1.2 Popularity Based Caching

The Popularity aware caching technique [ZSL05] use the access patterns of users for caching data segments with logs of the access patterns being maintained by a management server. The statistics gathered regarding user requests are used to determine the optimal number and placement of replicas for each individual video file. These popular segments are distributed among the peers participating in VoD sessions. As a result, popular data segments are easily obtained before playback. Each peer records its own seek operations information and sends it to the management server periodically, which can then perform the accounting functions on the basis of this information. The list of popular content elements is then distributed among peers. The scheduler of each peer requests for those popular content closest to its current playback position. The popularity aware caching techniques improves hit ratio by considering user's access patterns, however large computation are required to be performed by management server for extracting the list of popular content. The periodic playhead information with management server results in additional overhead.

6.2.1.3 Data Mining Based Caching

State maintenance and data mining [HL08] have also been proposed for acquiring more desirable segments of video. Each segment is identified by a unique segment number. Whenever a segment is played, its unique number is saved in a list by each peer. This playback history is exchanged among a set of peers (neighbouring peers) that share the closest play head positions. This playback history provides peers with a data set for performing data mining operations. Once a peer received the list of segment being played by neighbours (or peers in the same session), data mining is used to find the segments closely related to the current segments. Unlike popularity aware caching, each peer performs data mining operations locally instead of central management server. For that purpose, association rule mining is used to find maximum occurring segment with respect to current position.

6.2.2 Long Term Caching

Although on-demand multimedia content like VoD shares many of the requirements of short-term live streaming caching, the asynchronous nature of its content requests relax some to its constraints. A VoD session can be considered, to a first approximation, as a complete file which is cached at the first request and delivered for the following similar requests. This caching method has the following characteristics:

- Ingestion can be made in one shot or in multiple segments. Thus, the storage is easier and the memory management is not so complex but the storage volume could be very important.

- Cached segments could be chosen by cache to permit an overview of the content (i.e. trailer of a movie).
- Delivery is delay independent, VCR functions as seek mode is possible.
- User requests are first served by local cache. If this fails, the request is served by nearby caches. Finally, if these both fail, the request is then served by origin server.

6.2.3 Explicit vs. Transparent Caching

Content caching in P2P networks allows saving network resources and improves user experience. In addition, P2P caching mechanisms can be deployed seamlessly to clients, and without modifying the P2P protocol neither adding a new signalling mechanism. P2P caches allow bandwidth saving in the network by temporarily storing the more frequently requested content and distributing it directly to the peers. This section describes two different methods for caching at the network layer: explicit caching and transparent caching. These two methods try to achieve the same goal by using two different approaches: in-band and out of band approaches [OVERSI07]. In transparent caching both content consumer and content source are unaware of the presence of the cache, while in non-transparent (or explicit) caching one or both sides are aware of the presence of the cache.

With transparent caching, the network will transparently redirect P2P traffic to the cache, which will either serve the content directly or delegate the request to a remote P2P user while simultaneously caching the data. Transparency is typically implemented using Deep Packet Inspection (DPI), which identifies and directs P2P traffic to the P2P caching system for caching and request acceleration. Policy Based Routing (PBR) is another solution to divert the requested traffic to the transparent cache. PBR is a technique used in network routers to take routing decisions based on policies defined by network administrator. By using policy-based routing, network administrator can implement policies that selectively cause packets to take different paths. For example, PBR can be used to redirect specific traffic (e.g. HTTP, FTP) to a cache engine: all packets on TCP port number equal to 80 do not follow the classic route but are diverted to another router interface. PBR can be based also on source or destination IP addresses or other information contained in packet header.

Transparent caching offers multiple advantages:

- No client configuration is required, thus it reduces significantly network administration tasks.
- Traffic is automatically rerouted: users cannot bypass the cache service.
- The service is fail-safe, since the switch (or router) can bypass the cache server if it should happen to be down.
- The architecture is scalable.
- Load balancing between cache servers can be applied.
- In case of Deep Packet Inspection usage, the DPI device doesn’t have to give information which can reveal its presence and, consequently, cache presence. DPI is based on L2 network equipments, receiving and transmitting packets without any (or light) latency.

There are also some shortcomings related to this technique:

- The caching solution needs to analyse the two directions of the traffic, data sent by peers and data received by peers. These two ways of traffic are named upload and download traffic: upload means traffic sent, download means traffic received.
- Encrypted P2P protocols are not supported.

Non-transparent caching (explicit caching) is an out of band caching technique because the client side application knows the existence of the cache. In this mode, caching nodes are similar to

traditional real peers included in P2P overlay network. So, the cache follows the three stages of classical P2P behaviour on the network:

- Content discovery;
- Content acquisition;
- Content sharing.

There are many advantages of explicit caching:

- The failure of the system has no impact on users' traffic. There is no detection and no P2P redirection as the system works like a super peer.
- Being an integral part of the P2P network, this technology supports encrypted protocols.
- Out of band caching allows high performances in terms of P2P activity monitored. Load balancers are not useful.
- The capacity is easily upgradable to takes advantage of cumulative storage.
- The traffic is handled in a central location and requires no additional equipment or configuration changes to the existing network.

More details about explicit caching are given in details in the ENVISION deliverable D3.1 [D3.1].

6.3 Caching Requirements

Caching techniques are concerned by the following issues:

- Content ingestion: the ingestion of content to cache should follow well-defined criteria (content popularity, history based, etc.).
- Cached content coherence: All peers in the system should split a given set of content into the same set of chunks using a well-defined algorithm.
- Content storage should take into consideration content validity, timeliness, type (segments, chunks), and size.
- Distributed content delivery: segments must be delivered from different cache nodes.
- The role that different content caches will play in the network architecture should be clear. Two options have been considered concerning criteria for cache placement:
    - To give the best efficiency in term of QoE, cache should be placed as close as possible to end-user (in access network),
    - To get the best hit ratio (i.e. ingest a maximum of popular content), cache must “see” maximum of end-users requests (in core network).

6.4 ENVISION Caching Specification

6.4.1 ENVISION Cooperative Short-Term Caching

In our proposed cooperative caching technique ([AA09]), peers sharing the same session will periodically exchange their buffer map of available segments and their current play head position. For the management of the buffer, each peer caches the latest few minutes of the video played. Apart from this, each peer also holds the initial few minutes of video and never replaces this part during its existence in the network. This is due to the “impatient” behaviour of audience which scans through the beginning of videos to quickly determine their interest. The format of the buffer map is *[PeerID, Playback segments, Current Playhead, Timestamp],* where *PeerID* is the peer’s IP address, Playback segments refer to the record of segments the peer plays after it generates the last state-

messages. Instead of exchanging the complete record of available segment, each peer only sends *playback segments* in order to avoid extra overhead. *Timestamp* is the time when peer sends the state information. On receiving a state message, a peer performs relevant operations before it forwards the message to its neighbours. If peer 1 receives a state information message from peer 2, it compares the timestamp of current message with earlier timestamp. If the current timestamp is greater, then the state information record is updated. Once the state information is collected from all peers (in same session) each peer creates a table of available segments in that particular session. Each peer performs the necessary computation to remove redundancy and creates a list of available non-redundant segments in the session. Figure 41 illustrates the way peers exchange their buffer map and how they cooperate together to determine which segments is missing and which segments should be requested or not.

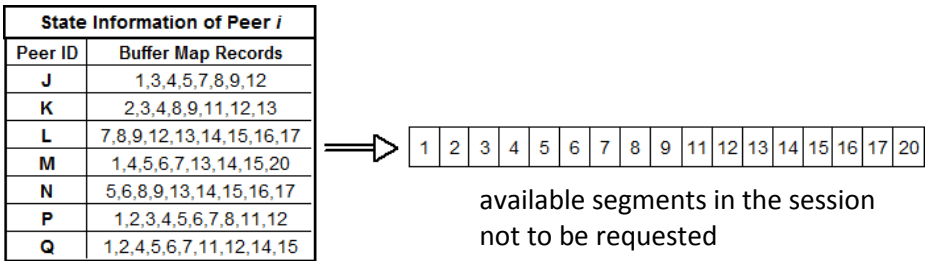


Figure 41: Mechanism for Cooperative Short-term Caching

The peer, then requests for a segment near to its play head position, which did not exist in that session. This request is made to either far neighbours or server (if there is no response from other peers). As a result, those rare segments are obtained from other session that did not exist in the current session. Later on, if a seek operation is carried out and the segment is available in the same session, it will take less time to acquire it from neighbour peers instead of server or far peers.

6.4.2 ENVISION Cooperative Long-Term Caching

The peers using ENVISION application can share long-term content like VoD or other type of files. In that case, the difference between long and short-term is probably that prefetching is not necessary.

The exchange of content between peers inside ENVISION overlay network use real network of several network services providers (i.e. autonomous system or AS). To accelerate download speed for each peer that requests the same content, ENVISION can collaborate with caching solution located in background network.

Network caching solution must first ingest the content (partially or completely). Content sharing between peers are analysed by the cache which is able to monitor ENVISION traffic. The cache stores the content: pieces of files or chunks or complete files following used protocol. For P2P protocol, pieces of content are treated as single files with a specific ID like chunks. In case of HTTP protocol, depending on cache mechanism, ingestion can be based on a divider which creates chunks (like P2P) or based on asynchronous ingestion for entire content. Once the piece, chunk or entire content is stored in the cache, next request for the same content will be served by the cache. In case of partial storage, the cache will serve the available pieces and ENVISION peer retrieve the other pieces directly from other ENVISION peers.

The distributed caches are also considered as network caching mechanisms which store the content at different locations to provide a high hit ratio following popularity of requests and content.

7. MULTILINK ENABLED PEERS FOR BOOSTING CONTENT DELIVERY

7.1 Introduction

This section introduces the concept of a multilink enabled peer (MLEP) that makes use of several networks simultaneously for receiving or sending data. The MLEP will be studied and developed to meet requirements for distribution of live video over P2P networks, making benefit of the CINA interface. The MLEP is a cross workpackage (WP) effort, as such in WP3, the mechanisms required by the MLEP from the network to improve delivery are defined, including multilink extensions to the CINA interface, multilink load balancing and SLA (Service Level Agreement) options for managing cross-network agreements. In WP4, techniques for content scheduling over multiple links are being investigated including the gathering of network information from multiple ISPs. This section focussed on the WP5-related aspects identifying how the MLEP will make use of multilink aware techniques for content adaptation, content protection and link aggregation.

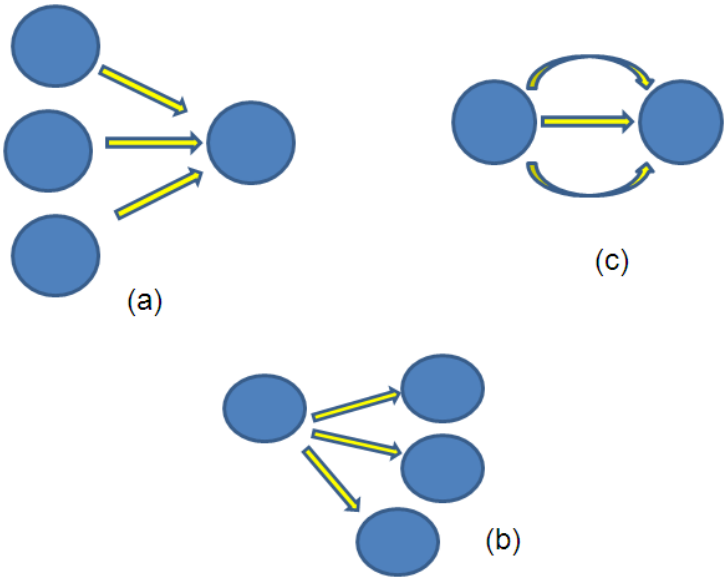


Figure 42: Multilink Illustrations

Figure 46 shows an overview of three multilink cases under consideration. The blue circles represent distinct nodes in the overlay while the arrows show connections between those nodes that are using distinct network interfaces, which may be to wired as well as wireless access networks. Note that multiple IP flows between a pair of nodes may exist over the same physical network interface, but such multiple flows are not shown in the figure. Figure 46(a) shows a MLEP receiving data from three different peers over three separate network interfaces, Figure 46(b) shows a MLEP transmitting data to three separate peers over three distinct network interfaces, while Figure 46(c) shows two MLEPs exchanging data between themselves over three network interfaces. Options (b) and (c) are relevant for content sources, for example video streams being generated in the field, which wish to make use of several available wireless networks (3G, Wi-Fi, WiMax, etc.) to maximise upload capacity and therefore video quality. Option (c) shows the case where the video stream is split into multiple sub-streams which are transmitted in parallel to a single aggregation node which may act as a proxy for the mobile video source to inject the full stream into the content distribution overlay. Option (b), on the other hand, shows the case where the three receiving peers inject the substreams they receive into the overlay swarm and if any one of them wishes to receive the full stream (to reproduce all layers of the original video encoding and render it with full quality) then they will need to exchange substreams with one another. There may also be combinations of the above three scenarios, e.g. (a) and (b) where the MLEP is acting as a relay of the stream in addition to, optionally, being a consumer.

The MLEP combines the following three technologies:

- Adaptive Live Video Streaming
- Multilink communication
- P2P communication

The MLEP will make use of the CINA interface to improve its services over the associated networks.

7.1.1 MLEP Adaptive Live Video Streaming

High quality live video streams may be required to be delivered from anywhere and at any time. There are not many technology solutions, today, that can achieve this without pre-planning and deploying infrastructure in advance. Such planning may include the arrangement of a microwave vehicle to communicate through satellites to feed main TV channels, or the use of local dedicated infrastructure deployed covering an event area and connected to the wired infrastructure through regional gateways. In many cases feeding high quality live content from the field to the Internet is not possible without specially planned resources. The industry of live streaming over the Internet is increasing rapidly, sites like UStream, and others supplying thousands of live streams to the Internet audience for free. It is done by distributing the uplink received video stream over P2P and CDN networks. However, the current state of the art technologies could be improved with P2P and CDN technologies benefitting from closer collaboration with the video source. The MLEP adaptive live video solution will be designed to meet P2P requirements, and will enable and improve the liveness of video services from anywhere into the Internet through ENVISION-enabled applications.

7.1.2 Multilink Enabled Peer for P2P Delivery

In this work, we will examine the requirements for P2P enhancements, based on the multilink capabilities of the peer (smartphones have multiple embedded modems such as 3G, 2G, Wi-Fi, and more). The homogeneous and heterogeneous networks today are not collaborating in a way that allow smooth boosting of capacity (as defined in the MARCH project, see section 7.2.2), even though access network segments do belong to the same provider. When multiple network providers are involved the problem is even more complicated and the use of the CINA interface and the associated multi-network consolidated overlay view defined in ENVISOIN D4.1 could bridge the gaps.

7.2 State of the Art

Prior contributions in aggregating multiple links over heterogeneous wireless connections is mostly limited to academic studies and research projects such as MARCH as described in section 7.2.2. The use of multilink enabled peers for P2P communication is novel in itself and there are very few studies on the specific problems associated with scheduling P2P content over multiple network interfaces simultaneously. Moreover the collaboration of ISPs and Network providers through the use of the CINA interface for multilink enabled peers is a new topic not specifically considered in the literature and emerging standards such as ALTO. In this chapter we will provide some background regarding evolving multilink technologies.

7.2.1 3GPP SA2 23.861 Standard

This standard deals with the use of multiple networks for different services at the same time, i.e. voice and data sessions, as described in its identified use cases below. Two main scenarios are defined where the user equipment (UE) is connected via different access networks simultaneously, sending and receiving different IP flows through different access networks. The standard defined the following use cases in sections 7.2.1.1 and 7.2.1.2.

7.2.1.1 Use Case 1

Michael is at home where both 3GPP and non-3GPP access networks are available. As an example, the non-3GPP access may be a domestic Wi-Fi hotspot. Michael is accessing different services with different characteristics in terms of QoS requirements and bandwidth:

- a Video Telephony call (VT),
- a media file synchronisation (e.g. a podcast and downloading of a TV series),
- a non-conversational video streaming (e.g. IPTV), and
- a P2P download.

Some of these flows may be from the same application (e.g. the Video Telephony may be via a virtual private network tunnel). Based on operator's policies, the user's preferences and the characteristics of the application and the accesses, the IP flows are routed differently. As an example, the audio media (conversational voice) of the VT call and the video streaming are routed via 3GPP access, while the video media (conversational video (live streaming)) of the VT, the P2P download (best effort) and media file synchronisation are routed through the non-3GPP access. The scenario is depicted in Figure 43.

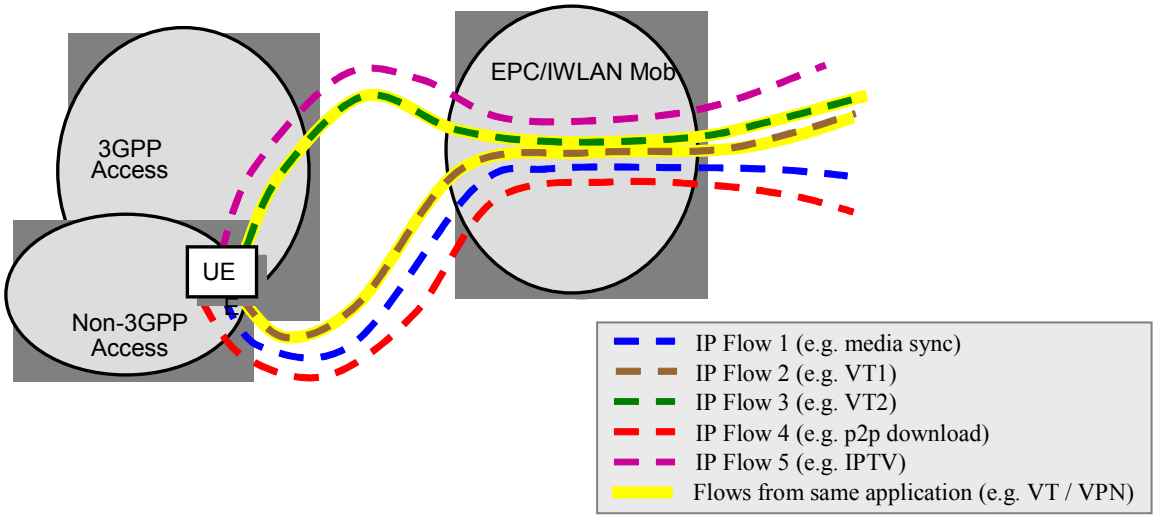


Figure 43: Routing of Different IP Flows through Different Accesses

After a while Michael moves out of the home and loses the non-3GPP connectivity. Triggered by this event, the IP flows need to be moved to the 3GPP access which is the only access available. Figure 44 shows how the IP flows are redistributed when the non-3GPP access is no longer available.

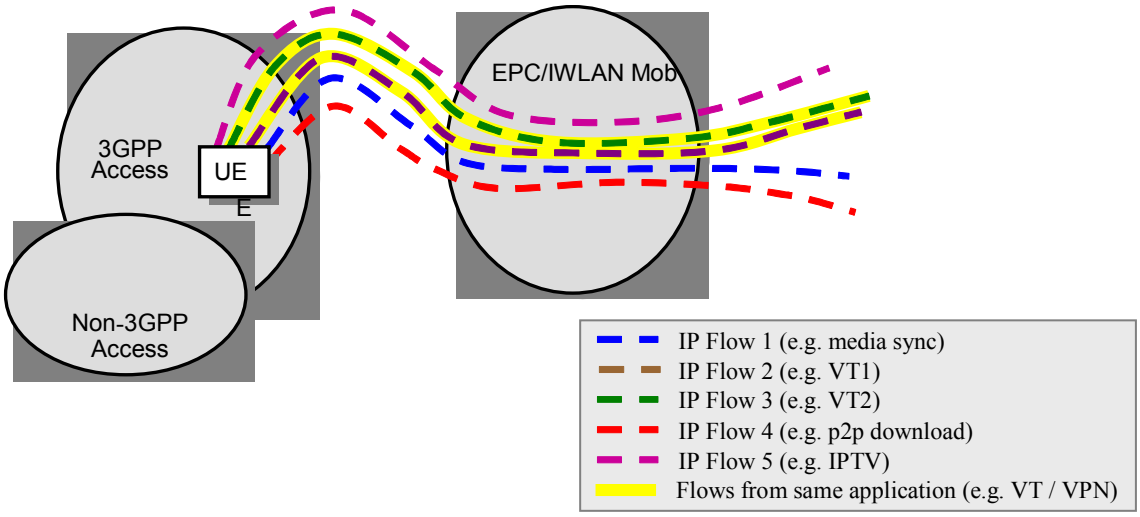


Figure 44: UE moves out from non-3GPP access and the IP flows are moved to 3GPP

Later on, Michael goes back home, or moves to another location where both 3GPP and non-3GPP connectivity are available. Triggered by this event, the video media of the VT, the P2P download and the media file synchronisation are moved back to the non-3GPP access. As a result, the scenario depicted in Figure 43 is restored.

7.2.1.2 Use Case 2

Michael is spending time by hanging out with an online friend through a multimedia session. They have a VoIP session (conversational voice) combined with video (conversational video). During the multimedia session Michael browses web (best effort) and occasionally watches cool video clips (Non conversational video). Based on the operator policy the VoIP flow and conversational video are routed via 3GPP access, while the non conversational video and best effort IP flows are routed via Non 3GPP. At 2am, Michael's device starts FTP file synchronisation with a backup server (best effort) (see Figure 45).

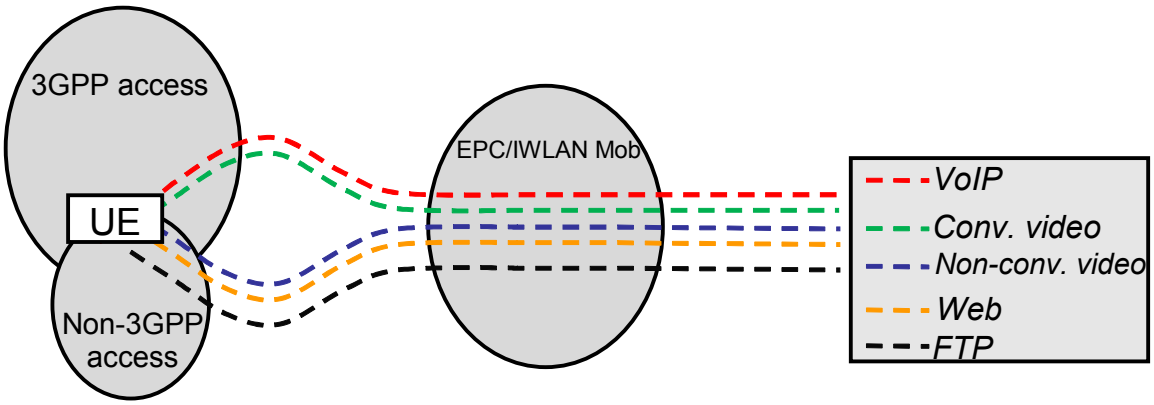


Figure 45: Splitting of IP Flows based on Operator's Policies

Due to the FTP synchronisation, the non 3GPP access becomes congested and the non conversational video flows are moved to 3GPP access (see Figure 46).

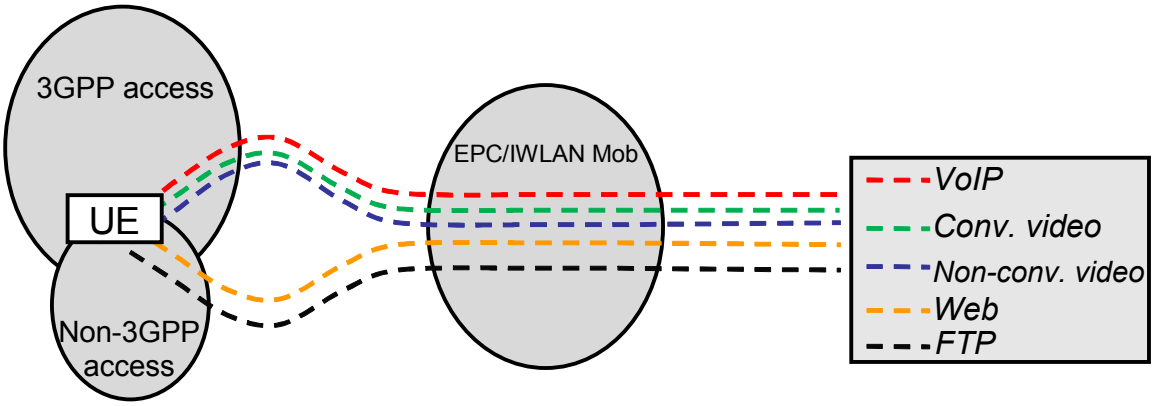


Figure 46: Movement of one IP Flow due to Network Congestion

Later as the HTTP server response time for the web browsing session (best effort) is detected to have increased, also the best effort web browsing was moved to the 3GPP access point. Only the FTP file synchronisation is left to non 3GPP access (see Figure 47).

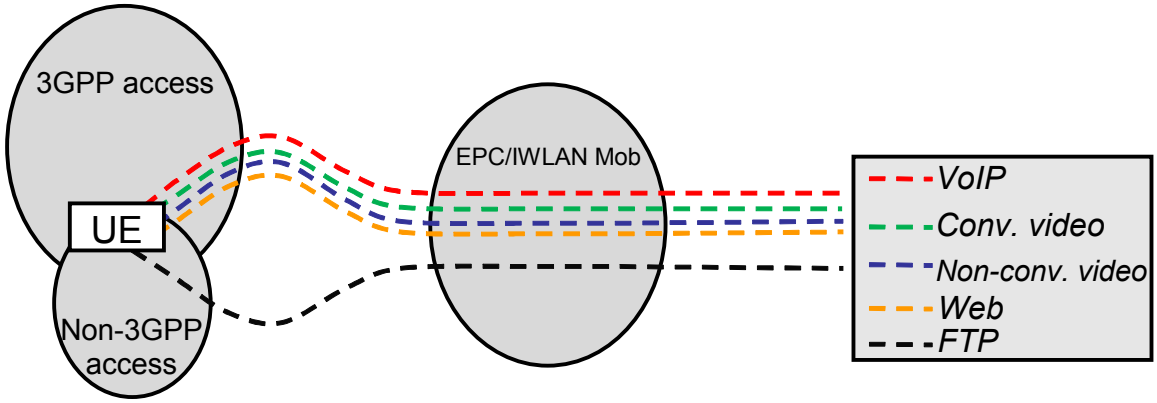


Figure 47: Further Movement of IP Flows due to Network Congestion

At 2:15am the FTP synchronisation completes and the non conversation video and Web browsing are moved back to non 3GPP access (see Figure 48).

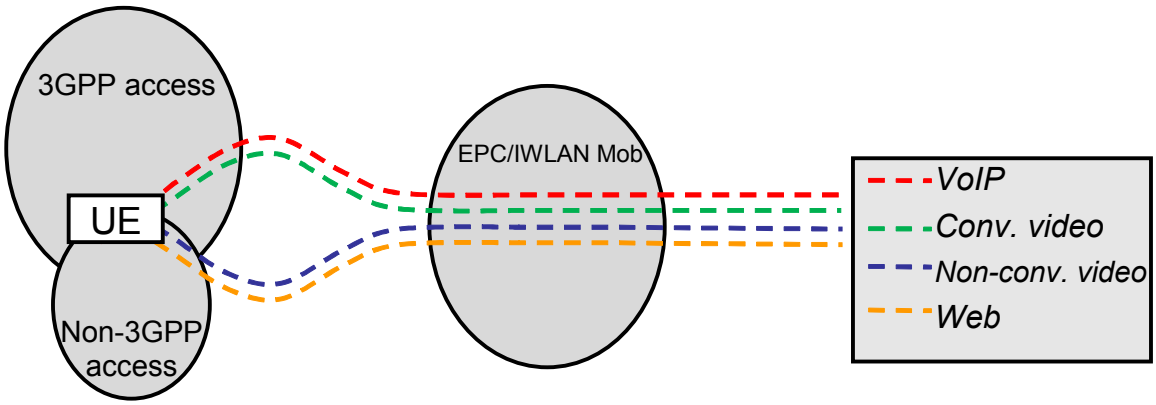


Figure 48: Distribution of IP Flows after Network Congestion is Over

7.2.2 The MARCH Project

The MARCH project is an ongoing Celtic project started in 2008 and ending in September 2011. This section introduces the MARCH multilink reference architecture.

7.2.2.1 Reference Architecture

The MARCH project proposes various use cases for multilink along with business model analysis, the most promising technology and use cases refer to collaboration between several wireless access networks for enabled devices. Figure 49 shows the proposed reference architecture for 3GPP networks.

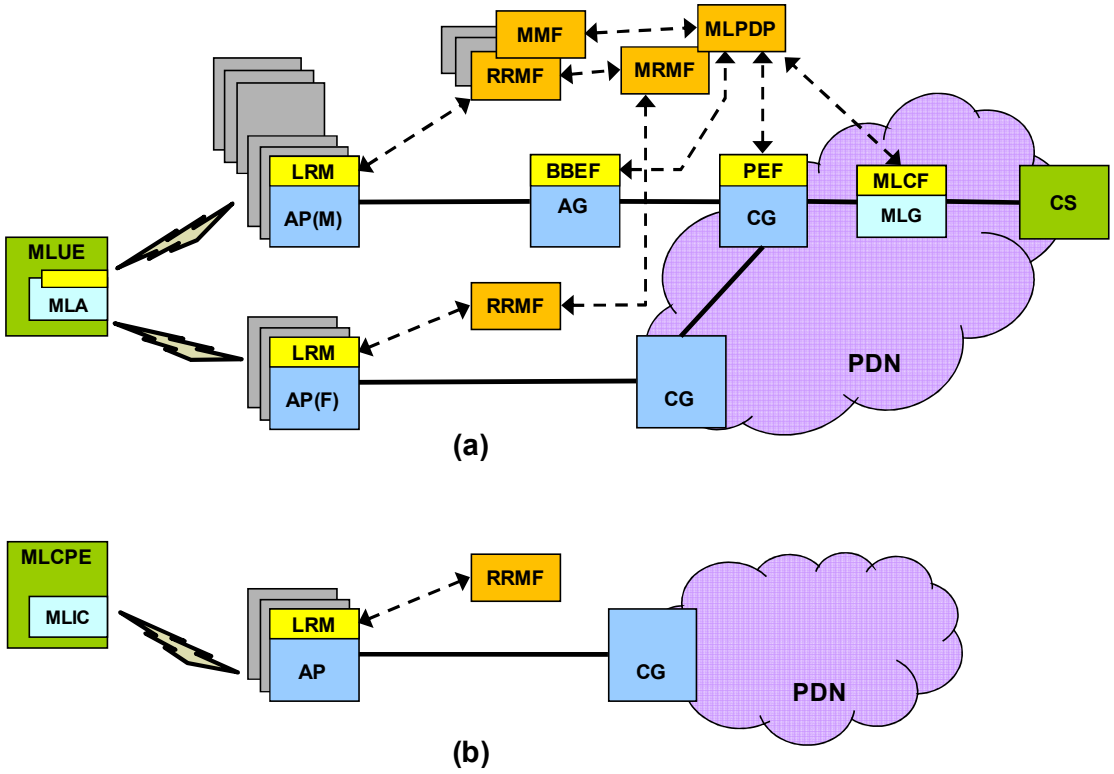


Figure 49: MARCH Multilink Reference Architecture

Figure 49(a) shows the reference architecture for network level (IP and above) multilink techniques. The multilink gateway (MLG) is applied in the case a single multilink service data flow is being split/merged. The MLG is deployed in the packet data network (PDN) to enable and support splitting/merging of a single media stream associated with end user equipment (UE) while allowing the routing of the media to occur as needed and decided by the network and/or the UE. The multilink adaptor (MLA) in the MLUE is similar to the MLG in its functionality as it also capable of splitting/merging of end user media streams, and performing content adaptation. The MLA controls function (yellow entity) is interacting with the Multilink Control Function of the MLG. For simplicity, this interaction is not shown. The ENVISION project may define additional interfaces between the overlay and the underlying networks to better share information and have mutual gain.

7.2.3 Multipath TCP (MPTCP)

TCP was designed when hosts were equipped with single interface, and only routers had several active network interfaces. As a result, TCP, as it is defined today, is limited to single path per connection. However, a lot has changed since TCP was invented, and today it is not rare to see a host device with multiple network interfaces. Every laptop has WiFi, Ethernet and sometimes even 3G interfaces. The proliferation of smart-phones equipped with both 3G and WiFi brings a growing number of multi-homed hosts to the network. It is clear that using multiple paths simultaneously will improve network resource usage and will provide better user experience in terms of higher throughput and improved network failures resilience.

IETF has formed a working group to develop and design Multipath TCP (MPTCP) protocol. MPTCP allows two hosts to use multiple paths when exchanging data over a single session. In general, MPTCP modifies existing TCP in such a way that a normal TCP socket is presented to the application layer, while in fact data are sent across several paths, where each path is called a subflow.

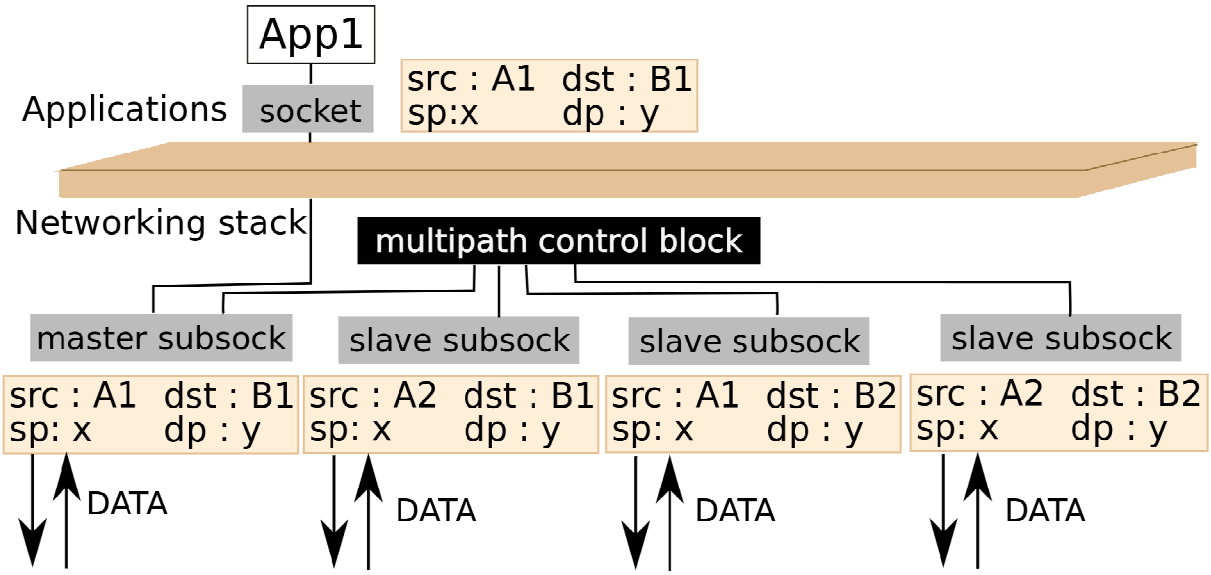


Figure 50: Linux MPTCP architecture

MPTCP flow example: Let's assume that a client with two addresses (A1 and A2) is trying to connect to a server with single address (B1):

1. Client establishes a TCP connection from A1 to B1 via standard three way handshake.
2. Once the first connection is established, client may establish another TCP connection, from A2 to B1.
3. Client will link the second connection to the first one by using MP_JOIN option, indicating the token of the first flow.
4. Both TCP connections are linked together into one multipath TCP session, and both can be used to send / receive data.

Since subflows may fail during the lifetime of a MPTCP session, a mechanism to retransmit data from one subflow on a different subflow is needed. Each subflow is equivalent to a regular TCP connection with its own 32bit sequence number, and MPTCP maintains an additional 64bit data sequence numbering space. This numbering allows the receiving side to reorder the packets in case it is received out-of-order, and to ask for retransmission in case that data is lost.

There are several implementations of MPTCP, one of them can be found at: http://mptcp.info.ucl.ac.be/ which is a MPTCP enabled server offering an installation package for Linux kernel patched to support MPTCP.

The Envision MLEP also uses multiple interfaces to boost the content delivery over the wireless access networks. The MPTCP is a candidate transport layer protocol to be used by the MLEP, although most likely the MLEP will be implemented using a similar approach over UDP based on a MPTCP friendly algorithm, and a FEC mechanism to handle loss or late arrival packets.

7.3 ENVISION Multilink Enabled Peer

7.3.1 Requirements

The ENVISION MLEP combines the three existing technologies of multi-link, P2P and live video streaming to be used as part of the live video delivery specified by the ENVISION use cases defined in D2.1 (bicycle race and Web3D conferencing). MLEPs will produce live video streams to be distributed over the P2P overlay swarm. The following requirements are identified:

- Real-time content generation should be encoded in H.264 SVC format.
- MLEP development shall be based on cross-layers optimisation to adapt encoding parameters according to the performance of the multiple links and the networks interconnecting the MLEP to its receiving peers, perform FEC, scheduling and monitoring.
- The MLEP techniques for content generation, adaptation, and protection will take into account the requirements identified in sections 2.3, 3.3. 4.3 and 5.3.
- The MLEP shall extend the use of multiple connections as described in 3GPP SA 23.861, to simultaneous distribution of a single stream over multiple connections and access networks.
- The objective is to enable high quality video to be delivered from the field through ENVISION-enabled distribution and swarming techniques, maximising QoE for the users.
- The MLEP will benefit from information gathered from the CINA interfaces with underlying ISPs as well as from overlay monitoring and overlay performance evaluation to: identify the available network capacity across all physical network interfaces; to optimise the peer selection process according to the accessibility of peers and associated performance metrics via different physical access networks; to determine the error protection levels at the content and transport levels; to balance the load across the available networks based on performance parameters as well as network costs signalled by the underlying ISPs via the CINA interface.
- The MLEP study will cover mechanisms for peer discovery through the consolidated overlay view provided in D4.1 by exposing the available interfaces rather than multiple independent discovery processes and more related aspects.

7.3.2 Initial Architecture

The initial architecture of the MLEP is shown in Figure 51, the architecture consists of modules related to WP3, WP4 and WP5 as follows:

- WP5: Content Encoding & Adaptation, and Content/Packet Error Protection
- WP4: Peer Selection, Data Scheduling, Monitoring & L7 Performance Evaluation
- WP3: Multiple Access Networks 1-N.

7.3.2.1 Traffic Flow

In Figure 51 the traffic flows (black arrows) from a video retrieval interface which capture the live content from the external device or camera, through the Content Encoding & Adaptation module and Content Error Protection. The Content Encoding & Adaptation block controls the video encoding attributes (see sections 3 and 4) at a bitrate that complies with the aggregate available resources over the multiple links and according to application layer performance estimation/evaluation. The Data Scheduling module is responsible for determining which encoded data is to be transmitted to which peers, and over which interfaces according to multi-link enhanced scheduling algorithms. Finally packet level protection may be optionally applied to protect the data flowing to a peer according to the specific characteristics of the communications path (error and loss rate, etc.) to the peer over the selected access network.

In the above it has been assumed that the MLEP is the content source, but some MLEPs may be consumers and/or relays of the video stream rather than sources in their own right. In these cases the content source will not be a locally attached capture device but will be formed from chunks received from other peers. Unless the node is additionally performing mid-point content adaptation, the Content Encoding & Adaptation and Content Error Protection blocks will not be active when relaying content, and the main multi-link-related functions in a MLEP in relaying mode will be multi-link data scheduling and optimisation.

7.3.2.2 Control Flow

The Peer Selection & Connection Establishment block receives requests for the delivery of content from interested peers. It will use information from the Consolidated Overlay View (see ENVISION D4.1) to select the best peers, based on the characteristics of the underlying network (delay, estimated throughput, etc). Performance estimates to the same peer may differ according to the access network the MLEP may use to initiate a connection to the remote node and therefore the peer selection process will need to be multi-link aware. Once peers have been selected and connections have been established Network Monitoring estimates link and network conditions and L7 Performance Evaluation computes application layer performance of the receiving peers and the swarm in general in terms of, for example, the SVC layers received within playout time threshold limits and the resulting spatial and temporal resolution of the video. This information will be used to adjust the encoding and adaptation parameters, the level of content protection, the scheduling of data between receiving peers over which network links, as well as the degree of transport layer packet protection per connection per access network. Content Encoding & Adaptation provides additional control information regarding the importance of the encoded content, which, when combined with monitoring information is input to the content error protection and data scheduling algorithms.

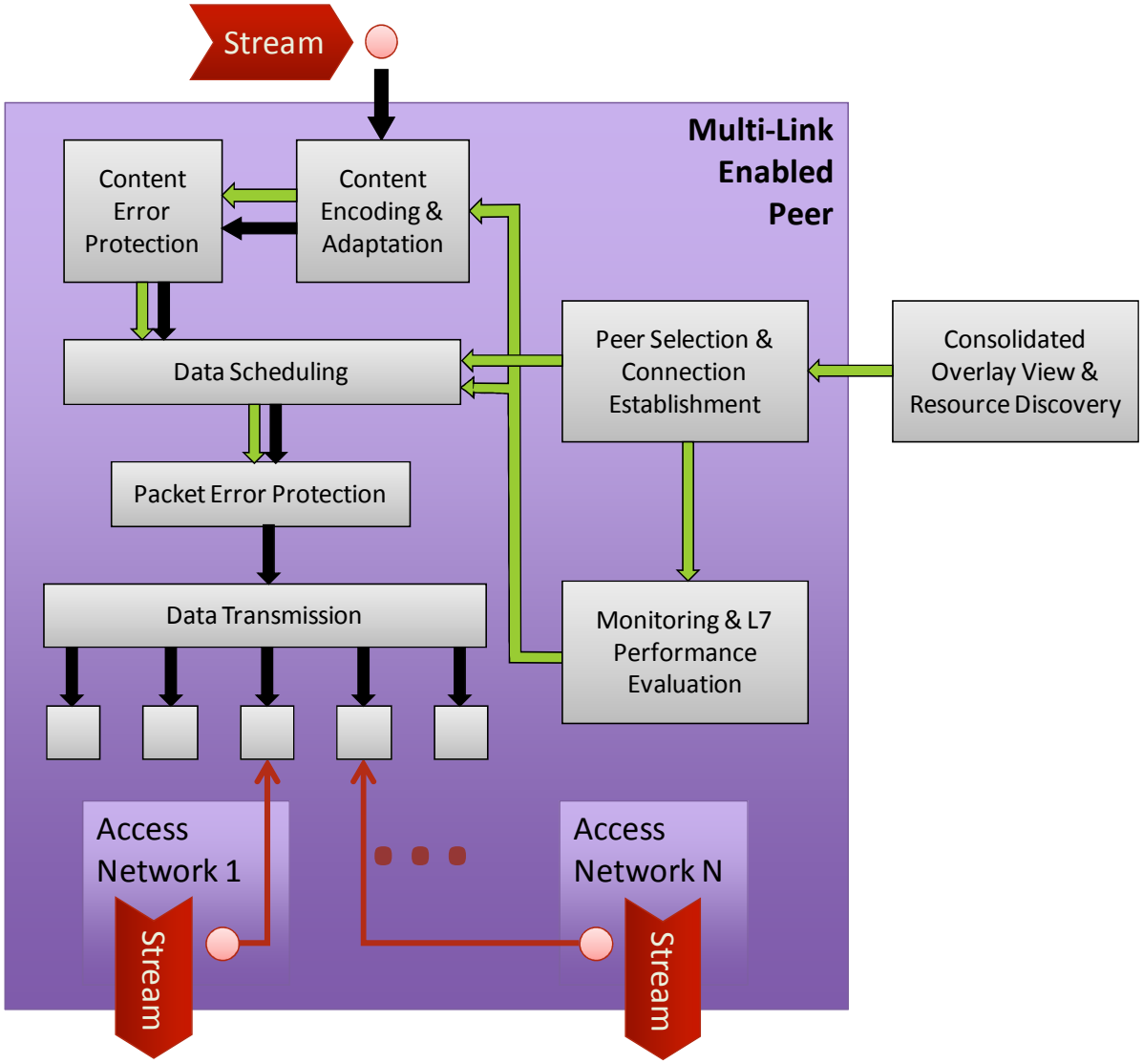


Figure 51: MLEP Initial Architecture

8. CONCLUSIONS

In this deliverable we investigated the current research trends in content generation and adaptation that take into consideration the capabilities of end-users and the changing conditions of the networks used for content delivery. Having specified content adaptation requirements for ENVISION, we presented initial specifications for metadata management, content generation and adaptation, error resilience, caching and multilink enabled transmission.

In a media distribution overlay, the metadata model plays an important role in the definition of a functional content distribution chains. Thus, the definition of a flexible metadata model is important for the eventual adoption of any media distribution overlay. In this deliverable we propose an initial metadata model for ENVISION. This model organises metadata into seven different classes (End User, Terminal, Content, Network, Service, Session and Peer). There are meant to describe the main actors and roles required for the specification of content adaptation functions.

With regards to content generation, we reviewed various candidate video encoding standards that include the widely used MPEG family, H.264, H.264/SVC and some emerging ones like VP8, WebM, and VC-1. These standards will be further investigated for possible exploitation in ENVISION. In addition, we presented a rich taxonomy for content adaptation techniques and an initial specification for the ENVISION content adaptation architecture. A related issue is that of media compression and the transmitted of compressed content over error prone networks, which requires efficient schemes based on source and channel models. In this regard, we presented a detailed analysis of the role of FEC in supporting live video delivery over P2P networks, at both the application and transport layers.

With regards to stored content, content caching can drastically increase network efficiency and lower delivery delay. We discussed several existing caching techniques, both at the network and overlay layers, and proposed a cooperative caching scheme for ENVISION. Finally, we described the MLEP mechanism, a scheme for boosting peer upload capacity that makes use of several networks simultaneously for receiving or sending data. This is of particular interest to mobile HD sources, since live video stream with high quality is often required to be delivered from anywhere at any given time. The MLEP will be studied in depth and further developed to meet the requirements of live video distribution over P2P networks, making use of the functionality provided by the CINA interface.

The future work in the context of ENVISION includes the refinement of the metadata model (user, terminal, network, content and service profiles) used in the content generation, consumption and adaptation processes. Moreover, we intend to design and implement the specified content adaptation functions to the extent that it is necessary to support the proposed use cases.

REFERENCES

[3GP07] http://www.openiptvforum.org/docs/Release2/OIPF-T1-R2-Specification-Volume-2a-HTTP-Adaptive-Streaming-V2_0-2010-09-07.pdf

[AA09] U. Abbasi and T. Ahmed, COOCHING: cooperative prefetching strategy for P2P video-on-demand system, in Lecture Notes in Computer Science, Wired-Wireless Multimedia Networks and Services Management, vol. 5842, pp. 195–200. Springer, Berlin (2009)

[ABI01] P. Amon, G. Bäse, K. Illgner, and J. Pandel, Efficient Coding of Synchronized H.26L Streams, ITU-T Video Coding Experts Group (VCEG), document VCEG-N35, Santa Barbara, CA, USA, September 2001.

[ACD04] T. Anker, R. Cohen, and D. Dolev. Transport Layer End-to-End Error Correcting. Technical report, The School of Computer Science and Engineering, Hebrew University, 2004.

[AD06] T. Ahmed, I. Djama, Delivering Audiovisual Content with MPEG-21-Enabled Cross-Layer QoS Adaptation in Packet Video 2006 published Springer-Verlag GmbH part of Journal of Zhejiang Univ SCIENCE A (IEEE Packet Video PV), Volume 7, Number 5, pp. 784-793. April 2006.

[AMK98] E. Amir, S. McCanne, and R. Katz, An Active Service Framework and its application to real-time multimedia transcoding, in SIGCOMM, symposium on communications architectures and protocols, September 1998.

[AMZ95] E. Amir, S. McCanne, and H. Zhang, an Application Level Video Gateway, In Proc. ACM Multimedia 1995.

[AP03] P. Amon, and J. Pandel, Evaluation of Adaptive and Reliable Video Transmission Technologies, Packet Video'03, France, April 2003.

[B03] I. Burnett et al., MPEG-21: Goals and Achievements, IEEE MultiMedia, vol. 10, no. 6, Oct.–Dec. 2003, pp. 60-70.

[BNZ08] S. Borade, B. Nakiboğlu, and L. Zheng, "Some fundamental limits of unequal error protection," in Proceedings of the IEEE International Symposium on Information Theory (ISIT '08), pp. 2222–2226, July 2008.

[BPM98] T. Bray, J. Paoli, C. M. Sperberg-McQueen, Extensible markup language (XML) 1.0 W3C recommendation, Technical report, W3C, 1998.

[CHG05] J. Chakareski, S. Han, and B. Girod, Layered coding vs. multiple descriptions for video streaming over multiple paths, in Multimedia Systems, Springer, online journal publication: Digital Object Identifier (DOI) 10.1007/s00530-004-0162-3, January 2005.

[CJL07] B. Cheng, H. Jin and X. Liao, Supporting VCR functions in P2P VoD services using ring-assisted overlays, In Proc of ICC 2007.

[CLK09] Y. C. Chang, S. W. Lee, R. Komiya, Recent Advances in Transport Level Error Control Techniques for Wireless Video Transmission, International Conference on Multimedia, Signal Processing and Communication Technologies (IMPACT-2009), pp. 86-90, 14-16 March 2009, Aligarh, India.

[CSP01] S. F. Chang, T. Sikora, and A. Puri, Overview of MPEG-7 Standard, IEEE Trans. Circuits Syst. Video Technol., vol. 11, pp. 688–695, June 2001

[CV05] S. F. Chang and A. Vetro, Video adaptation: Concepts, technologies and open issues, Proc. IEEE—Special Issue on Advances in Video Coding and Delivery, vol. 93, no. 1, pp. 148–158, Jan. 2005.

[D2.1] ENVISION deliverable D2.1, Final Specification of Use Cases, Requirements, Business Models and the System Architecture, January 2011, FP7 ICT ENVISION project, www.envision-project.org

[D3.1] ENVISION deliverable D3.1, Initial Specification of the ENVISION Interface, Network Monitoring and Network Optimisation Functions, January 2011, FP7 ICT ENVISION project, www.envision-project.org

[D4.1] ENVISION deliverable D4.1, Initial Specification of Consolidated Overlay View, Data Management Infrastructure, Resource Optimisation and Content Distribution Functions, January 2011, FP7 ICT ENVISION project, www.envision-project.org

[DANB08] I. Djama, T. Ahmed, A. Nafaa and R. Boutaba, "Meet In the Middle Cross-Layer Adaptation for Audiovisual Content Delivery" published in the IEEE Transactions on Multimedia, Volume 10, Issue 1, pp. 105 – 120. Jan. 2008.

[DTH05] S. Devillers, C. Timmerer, J. Heuer, and H. Hellwagner, Bitstream Syntax Description-Based Adaptation in Streaming and Constrained Environments, IEEE Transactions on Multimedia, vol. 7, no. 3, pp. 463- 470, June 2005.

[FC01] K. French and M. Claypool, Repair of Streaming Multimedia with Adaptive Forward Error Correction, Proceedings of SPIE Multimedia Systems and Applications (part of ITCom), Denver, Colorado, USA, August 2001

[FHK06] P. Fröjdh, U. Horn, M. Kampmann, A. Nohlgren, and M. Westerlund, adaptive Streaming within the 3GPP Packet-Switched Streaming Service

[FTS] http://www.microsoft.com/casestudies/Case_Study_Detail.aspx?casestudyid= 400000721

[HAP05] A. Hutter, P. Amon, G. Panis, E. Delfosse, M. Ransburg, H. Hellwagner, Automatic adaptation of streaming multimedia content in a dynamic and distributed environment, IEEE International Conference on Image Processing (ICIP 2005), vol. 3, pp. 716-719, September 2005.

[HI98] J. Hunter and R. Iannella, The Application of Metadata Standards to Video Indexing, Second European Conference on Research and Advanced Technology for Digital Libraries, Crete, Greece, September 1998.

[HL08] Y. He, Y. Liu, VOVO: VCR-Oriented Video-on-Demand in Large-Scale P2P Networks, in Proc of IEEE Trans, parallel and Distributed Systems vol. PP, Issue 99, June.2008.

[HPN97] B. Haskell, A. Puri, and A. Netravali , Digital video: an introduction to MPEG, ISBN 0-412-08411-2, 1997

[I-SHARE] http://www.freeband.nl/project.cfm?language=en&id=520)

[ISO3166] http://www.iso.org/iso/country_codes.htm

[ISO639] http://www.w3.org/WAI/ER/IG/ert/iso639.htm

[KA03] N. Kamaci, Y. Altunbask, Performance comparison of the emerging H.264 video coding standard with the existing standards, in IEEE Int. Conf. Multimedia and Expo, ICME'03, Baltimore, MD, July 2003

[KPZ10] R. Kazhamiakin, M. Pistore and A. Zengin, Cross-Layer Adaptation and Monitoring of Service-Based Applications, ICSOC/ServiceWave 2009 Workshops, LNCS 2010.

[KK05] V. Kawadia and P. R. Kumar, A Cautionary Perspective on Cross Layer Design, IEEE Wireless Communications, vol. 12, no. 1, pp. 3- 11, February 2005

[KTK10] H. H. Kenchannavar, S. Thomas, and U. P. Kulkarni, Efficiency of FEC coding in IP networks. In Proceedings of the International Conference and Workshop on Emerging Trends in Technology (ICWET '10), 2010. ACM, New York, NY, USA, 358-361

[LSP07] Z. Liu, Y. Shen, S. Panwar, et al., Efficient Substream Encoding and Transmission for P2P Video on Demand, Packet Video 2007, Nov. 2007, pp. 143-152.

[MDK04] D. Mukherjee, E. Delfosse, J.-G. Kim, Y. Wang, Terminal and Network Quality of Service, invited paper, IEEE Trans. On Multimedia Special Issue on MPEG-21, 2004

[MMA02] N. Mikael, P. Matthias, NAEVE Ambjörn, Semantic Web Meta-data for e-Learning, Some Architectural Guidelines, Proceedings of the 11th World Wide Web Conference, Hawaii, 2002

[MSS02] B.S. Manjunath, P. Salembier, and T. Sikora, editors, Introduction to MPEG–7: Multimedia Content Description Interface, John Wiley & Sons Ltd., June 2002.

[NSM] http://www.iptc.org/std/NewsML/1.2/documentation/NewsML_1.2-doc-Guidelines_1.01.pdf

[OVERSI07] http://www.oversi.com/images/stories/white_paper_july.pdf

[PAN04] http://tools.ietf.org/html/draft-pantos-http-live-streaming-04

[PE02] F. C. N. Pereira, T. Ebrahimi, The MPEG-4 book, 2002, ISBN-0-13-061621-4

[PMPEG11] http://www.pro-mpeg.org/

[PR01] R. Puri and K. Ramchandran, Multiple description source coding through forward error correction codes, in 33rd Asilomar Conf. Signals, Systems and Computers, Oct. 1999

[PS00] S. Pfeier and U. Srinivasan, TV Anytime as an application scenario for MPEG-7, In Proc. of Workshop on Standards, Interoperability and Practice of the 8th International Conference on Multimedia, ACM Multimedia, Los Angeles, California, 2000

[R10] L. Richardson, The H.264 Advanced Video Compression Standard, 2010, ISBN-978-0-470-51692-8

[RAB02] T.S. Rappaport, A. Annamalai, R.M. Buehrer, and W.H. Tranter, Wireless communications: past events and a future perspective, IEEE Communications Magazine, vol. 40, no. 5, pp. 148-161, May 2002

[RFC1766] http://www.ietf.org/rfc/rfc1766.txt

[RFC2413] http://www.ietf.org/rfc/rfc2413.txt

[RS60] I. S. Reed and G. Solomon, Polynomial Codes Over Certain Finite Fields, SIAM Journal of Applied Math., vol. 8, 1960, pp. 300-304

[RTM09] http://www.adobe.com/devnet/rtmp/

[RTV06] P. Rubino, G. Tirilly and M. Varela, Evaluating users' satisfaction in packet networks using random neural networks. Proceedings of the 16th International Conference on Artificial Neural Networks (ICANN'06), September 2006

[RYH00] R. Rejaie, H. Yu, M. Handley, and D. Estrin, Multimedia proxy caching mechanism for quality adaptive streaming applications in the internet, in Proc of IEEE INFOCOM 2000

[SLP06] Y. Shen, Z. Liu, S. Panwar, K. Ross, Y. Wang, On the design of prefetching strategies in a peer-driven Video on demand systems, in Proc of IEEE International conference on Multimedia and Expo 2006

[SMW04] H. Schwarz, D. Marpe, and T. Wiegand, SNR-Scalable Extension of H.264/AVC, in proceedings of ICIP2004, Singapore, 2004

[SMW06] H. Schwarz, D. Marpe, and T. Wiegand, Overview of the Scalable H.264/MPEG4-AVC Extension, in Proc. IEEE International Conference on Image Processing, pp. 161-164, Atlanta, USA, Oct. 2006

[SMW07] H. Schwarz, D. Marpe, and T. Wiegand, Overview of the Scalable Video Coding Extension of the H.264/AVC Standard, IEEE Transactions on Circuits and Systems for Video Technology), 17(9):1103{1120, September 2007

[SSA09] http://alexzambelli.com/blog/2009/02/10/smooth-streaming-architecture/

[SVX07] H. Sun, A. Vetro, and J. Xin, An overview of scalable video streaming, Wireless Communications and Mobile Computing, vol. 7, no. 2, pp. 159-172, Feb. 2007

[VID] http://www.vidyo.com/documents/datasheets-brochures/vidyo_conferencing.pdf

[WAF99] S. Wee, J. Apostolopoulos and N. Feamster, Field-to-Frame Transcoding with Temporal and Spatial Downsampling, IEEE International Conference on Image Processing, October 1999

[WBSS04] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, “Image quality assessment: From error visibility to structural similarity,” IEEE Trans. Image Processing, vol. 13, no. 4, pp. 600-612, Apr.2004

[WHZ00] D. Wu, T. Hou and Y.-Q. Zhang, Scalable Video Coding and Transport over Broadband Wireless Networks, Proceedings of the IEEE, Sept. 2000

[Z09] T. Zahariadis, et al., “Content Adaptation Issues in the Future Internet”, in: G. Tselentis, et. al. (eds.), Towards the Future Internet, IOS Press, 2009, pp.283-292

[Z65] L.A. Zadeh. Fuzzy sets. Information and control, 8(3):338-353, June 1965. 2.1.3

[ZSL05] C. Zheng, G. Shen and S. Li, Distributed Prefetching Scheme for Random Seek Support in P2P Streaming Applications, In Proc of ACM workshop on Advances in P2P multimedia streaming 2005

APPENDIX A: Metadata classes XSD (XML Schema Definition)

a) End user

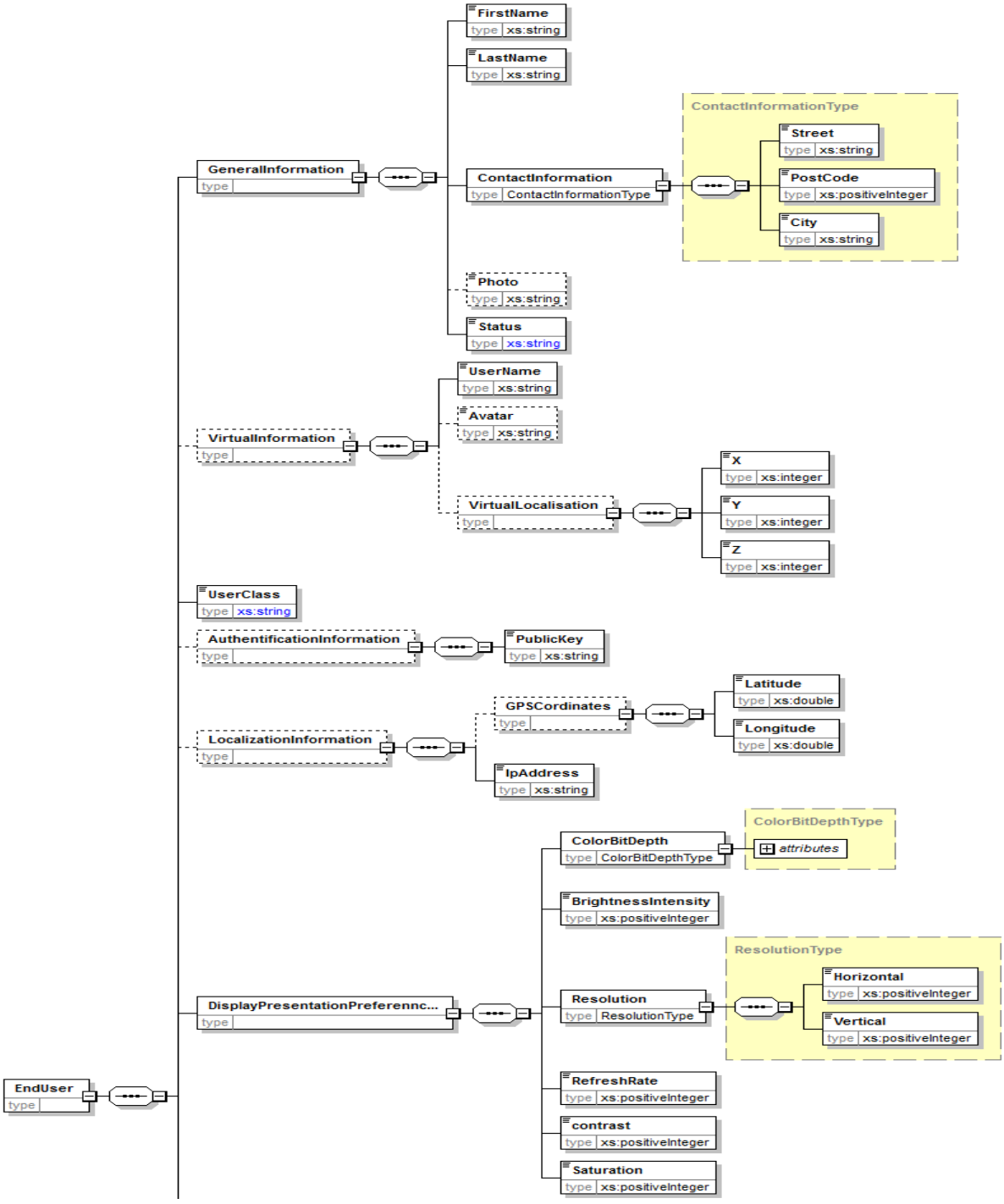
1. XSD code

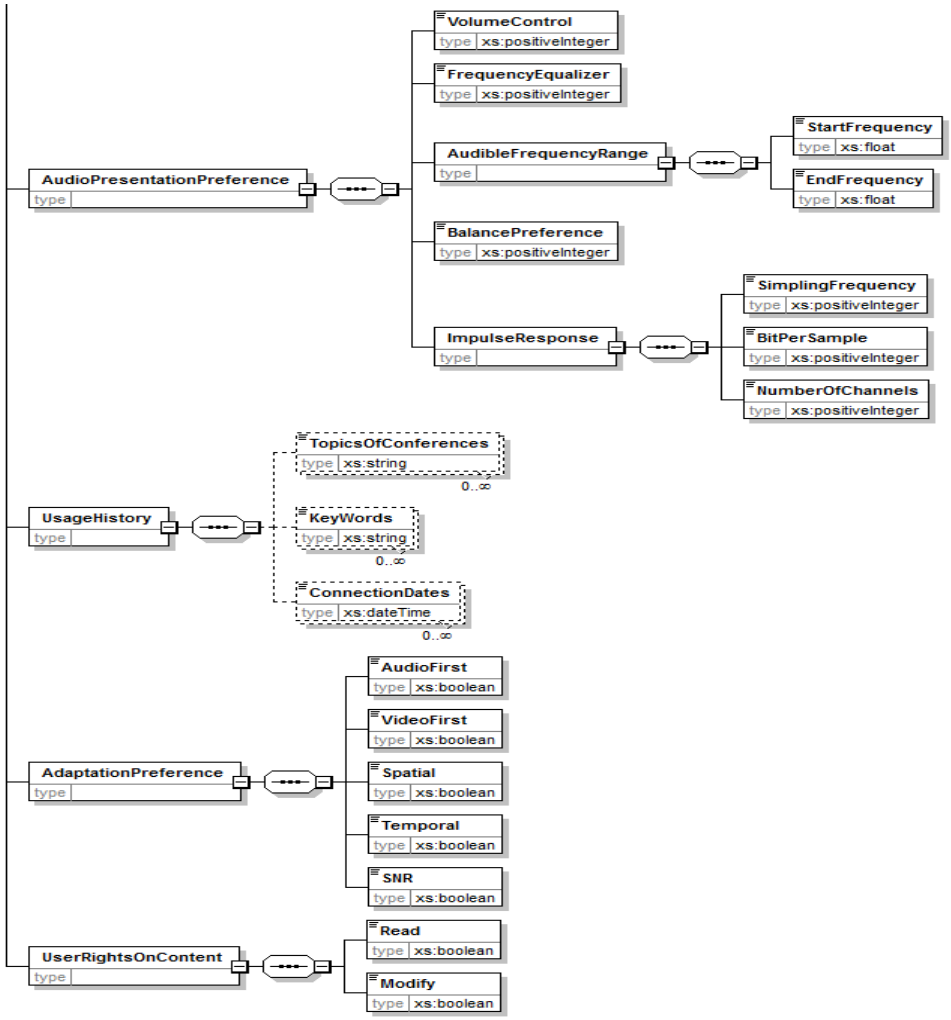
```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"   xmlns:ENVISION ="http://www.ENVISION-project.org/ENVISION"
elementFormDefault="qualified" attributeFormDefault="unqualified">
        <xs:include schemaLocation="ENVISIONTypesDeclaration.xsd"/>
        <!--#########################################################################-->
        <!--Describe the end user (a content consumer)-->
        <!--#########################################################################-->
        <xs:element name="EndUser"  >
                <xs:complexType>
                        <xs:sequence>
                                <xs:element name="GeneralInformation">
                                        <xs:complexType>
                                                <xs:sequence>
                                                        <xs:element name="FirstName" type="xs:string"/>
                                                        <xs:element name="LastName" type="xs:string"/>
                                                        <xs:element name="ContactInformation"
                                                                type="ContactInformationType"/>
                                                        <xs:element name="Photo" type="xs:string" minOccurs="0"/>
                                                        <xs:element name="Status">
                                                                <xs:simpleType>
                                                                        <xs:restriction base="xs:string">
                                                                                <xs:enumeration
                                                                                   value="Person"/>
                                                                                <xs:enumeration
                                                                                   value="Group of person"/>
                                                                                <xs:enumeration
                                                                                   value="Organization"/>
                                                                        </xs:restriction>
                                                                </xs:simpleType>
                                                        </xs:element>
                                                </xs:sequence>
                                        </xs:complexType>
                                </xs:element>
                                <xs:element name="VirtualInformation" minOccurs="0">
                                        <xs:complexType>
                                                <xs:sequence>
                                                        <xs:element name="UserName" type="xs:string"/>
                                                        <xs:element name="Avatar" type="xs:string" minOccurs="0"/>
                                                        <xs:element name="VirtualLocalisation" minOccurs="0">
                                                                <xs:complexType>
                                                                        <xs:sequence>
                                                                                <xs:element name="X"
                                                                                        type="xs:integer"/>
                                                                                <xs:element name="Y"
                                                                                        type="xs:integer"/>
                                                                                <xs:element name="Z"
                                                                                        type="xs:integer"/>
                                                                        </xs:sequence>
                                                                </xs:complexType>
                                                        </xs:element>
                                                </xs:sequence>
                                        </xs:complexType>
                                </xs:element>
                                <xs:element name="UserClass">
                                        <xs:simpleType>
                                                <xs:restriction base="xs:string">
                                                        <xs:enumeration value="Simple"/>
                                                        <xs:enumeration value="premium"/>
                                                </xs:restriction>
                                        </xs:simpleType>
                                </xs:element>
                                <xs:element name="AuthentificationInformation" minOccurs="0">
                                        <xs:complexType>
                                                <xs:sequence>
                                                        <xs:element name="PublicKey" type="xs:string"/>
                                                </xs:sequence>
                                        </xs:complexType>
                                </xs:element>
                                <xs:element name="LocalizationInformation" minOccurs="0">
                                        <xs:complexType>
                                                <xs:sequence>
                                                        <xs:element name="GPSCordinates" minOccurs="0">
                                                                <xs:complexType>
                                                                        <xs:sequence>
                                                                                <xs:element name="Latitude"
                                                                                        type="xs:double"/>
```

```
                                                                <xs:element
                                                            name="Longitude" type="xs:double"/>
                                                        </xs:sequence>
                                                    </xs:complexType>
                                                </xs:element>
                                                <xs:element name="IpAddress" type="xs:string"/>
                                            </xs:sequence>
                                        </xs:complexType>
                                    </xs:element>
                                    <xs:element name="DisplayPresentationPreferennces">
                                        <xs:complexType>
                                            <xs:sequence>
                                                <xs:element name="ColorBitDepth" type="ColorBitDepthType"/>
                                                <xs:element name="BrightnessIntensity"
                                                    type="xs:positiveInteger"/>
                                                <xs:element name="Resolution" type="ResolutionType"/>
                                                <xs:element name="RefreshRate" type="xs:positiveInteger"/>
                                                <xs:element name="contrast" type="xs:positiveInteger"/>
                                                <xs:element name="Saturation" type="xs:positiveInteger"/>
                                            </xs:sequence>
                                        </xs:complexType>
                                    </xs:element>
                                    <xs:element name="AudioPresentationPreference">
                                        <xs:complexType>
                                            <xs:sequence>
                                                <xs:element name="VolumeControl" type="xs:positiveInteger"/>
                                                <xs:element name="FrequencyEqualizer"
                                                    type="xs:positiveInteger"/>
                                                <xs:element name="AudibleFrequencyRange">
                                                    <xs:complexType>
                                                        <xs:sequence>
                                                            <xs:element
                                                        name="StartFrequency" type="xs:float"/>
                                                            <xs:element
                                                        name="EndFrequency" type="xs:float"/>
                                                        </xs:sequence>
                                                    </xs:complexType>
                                                </xs:element>
                                                <xs:element name="BalancePreference"
                                                    type="xs:positiveInteger"/>
                                                <xs:element name="ImpulseResponse">
                                                    <xs:complexType>
                                                        <xs:sequence>
                                                            <xs:element
                                                    name="SimplingFrequency" type="xs:positiveInteger"/>
                                                            <xs:element
                                                    name="BitPerSample" type="xs:positiveInteger"/>
                                                            <xs:element
                                                    name="NumberOfChannels" type="xs:positiveInteger"/>
                                                        </xs:sequence>
                                                    </xs:complexType>
                                                </xs:element>
                                            </xs:sequence>
                                        </xs:complexType>
                                    </xs:element>
                                    <xs:element name="UsageHistory">
                                        <xs:complexType>
                                            <xs:sequence>
                                                <xs:element name="TopicsOfConferences" type="xs:string"
                                                    minOccurs="0" maxOccurs="unbounded"/>
                                                <xs:element name="KeyWords" type="xs:string" minOccurs="0"
                                                    maxOccurs="unbounded"/>
                                                <xs:element name="ConnectionDates" type="xs:dateTime"
                                                    minOccurs="0" maxOccurs="unbounded"/>
                                            </xs:sequence>
                                        </xs:complexType>
                                    </xs:element>
                                    <xs:element name="AdaptationPreference">
                                        <xs:complexType>
                                            <xs:sequence>
                                                <xs:element name="AudioFirst" type="xs:boolean"/>
                                                <xs:element name="VideoFirst" type="xs:boolean"/>
                                                <xs:element name="Spatial" type="xs:boolean"/>
                                                <xs:element name="Temporal" type="xs:boolean"/>
                                                <xs:element name="SNR" type="xs:boolean"/>
                                            </xs:sequence>
                                        </xs:complexType>
                                    </xs:element>
                                    <xs:element name="UserRightsOnContent">
                                        <xs:complexType>
                                            <xs:sequence>
                                                <xs:element name="Read" type="xs:boolean"/>
                                                <xs:element name="Modify" type="xs:boolean"/>
                                            </xs:sequence>
                                        </xs:complexType>
                                    </xs:element>
```

```
                                </xs:sequence>
                            </xs:complexType>
                        </xs:element>
</xs:schema>
```

2. XSD visual format





b) Terminal capabilities

1. XSD code

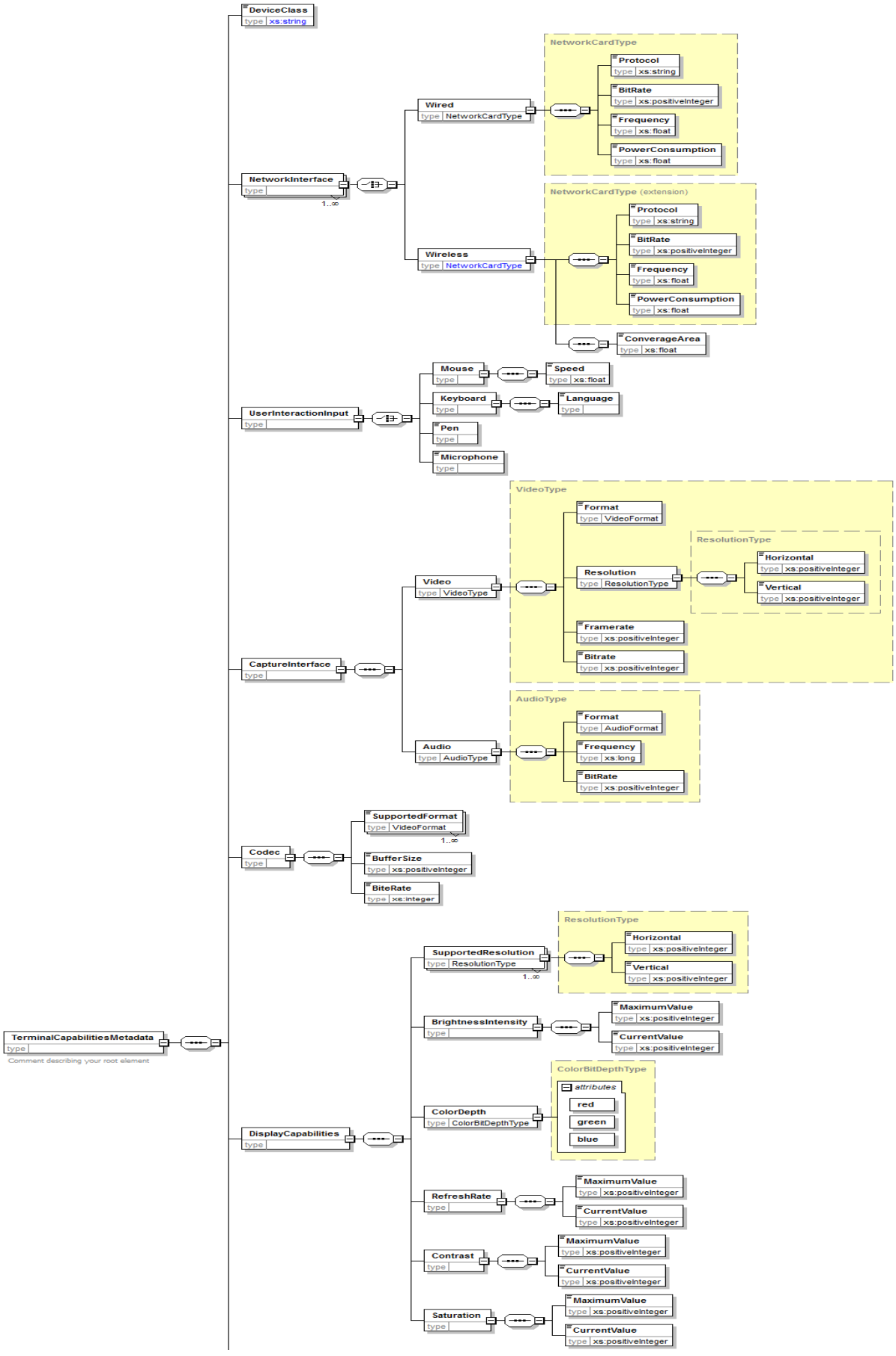
```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"   xmlns:ENVISION ="http://www.ENVISION-project.org/ENVISION"
elementFormDefault="qualified" attributeFormDefault="unqualified">
        <xs:include schemaLocation="ENVISIONTypesDeclaration.xsd"/>
        <xs:element name="TerminalCapabilitiesMetadata">
                <xs:annotation>
                        <xs:documentation>Comment describing your root element</xs:documentation>
                </xs:annotation>
                <xs:complexType>
                        <xs:sequence>
                                <xs:element name="DeviceClass">
                                        <xs:simpleType>
                                                <xs:restriction base="xs:string">
                                                        <xs:enumeration value="PC"/>
                                                        <xs:enumeration value="PDA"/>
                                                        <xs:enumeration value="Laptop"/>
                                                        <xs:enumeration value="Mobile phone"/>
                                                </xs:restriction>
                                        </xs:simpleType>
                                </xs:element>
                                <xs:element name="NetworkInterface" maxOccurs="unbounded">
                                        <xs:complexType>
                                                <xs:choice>
                                                        <xs:element name="Wired" type="NetworkCardType"/>
                                                        <xs:element name="Wireless">
                                                                <xs:complexType>
```

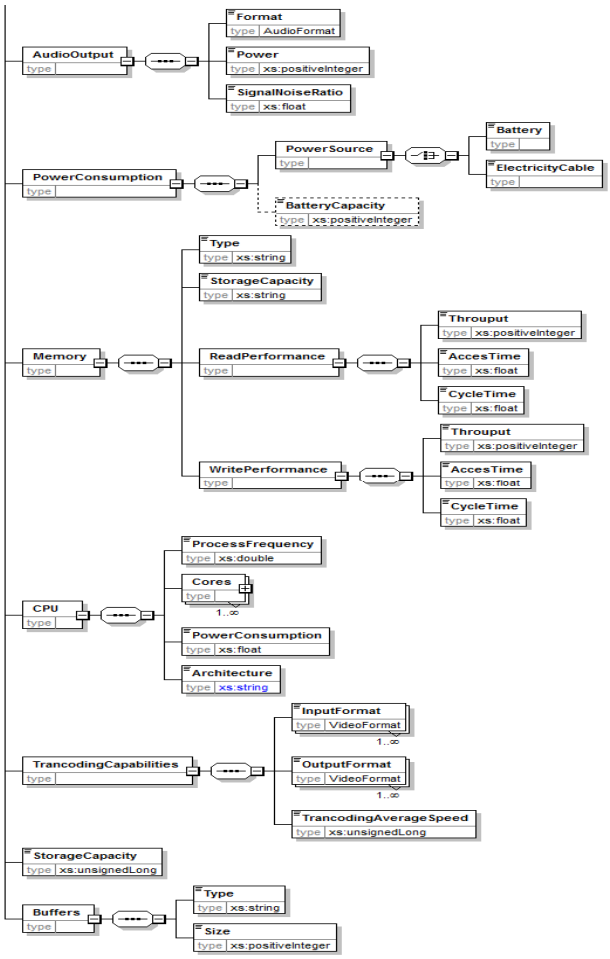
```
                                                <xs:complexContent>
                                                    <xs:extension
                                                      base="NetworkCardType">
                                                        <xs:sequence>
                                                        <xs:element
                                                  name="CoverageArea" type="xs:float"/>
                                                        </xs:sequence>
                                                    </xs:extension>
                                                </xs:complexContent>
                                            </xs:complexType>
                                        </xs:element>
                                    </xs:choice>
                                </xs:complexType>
                            </xs:element>
                            <xs:element name="UserInteractionInput">
                                <xs:complexType>
                                    <xs:choice>
                                        <xs:element name="Mouse">
                                            <xs:complexType>
                                                <xs:sequence>
                                                    <xs:element name="Speed"
                                                                type="xs:float"/>
                                                </xs:sequence>
                                            </xs:complexType>
                                        </xs:element>
                                        <xs:element name="Keyboard">
                                            <xs:complexType>
                                                <xs:sequence>
                                                    <xs:element
                                                        name="Language"/>
                                                </xs:sequence>
                                            </xs:complexType>
                                        </xs:element>
                                        <xs:element name="Pen"/>
                                        <xs:element name="Microphone"/>
                                    </xs:choice>
                                </xs:complexType>
                            </xs:element>
                            <xs:element name="CaptureInterface">
                                <xs:complexType>
                                    <xs:sequence>
                                        <xs:element name="Video" type="VideoType"/>
                                        <xs:element name="Audio" type="AudioType"/>
                                    </xs:sequence>
                                </xs:complexType>
                            </xs:element>
                            <xs:element name="Codec">
                                <xs:complexType>
                                    <xs:sequence>
                                        <xs:element name="SupportedFormat" type="VideoFormat"
                                                    maxOccurs="unbounded"/>
                                        <xs:element name="BufferSize" type="xs:positiveInteger"/>
                                        <xs:element name="BiteRate" type="xs:integer"/>
                                    </xs:sequence>
                                </xs:complexType>
                            </xs:element>
                            <xs:element name="DisplayCapabilities">
                                <xs:complexType>
                                    <xs:sequence>
                                        <xs:element name="SupportedResolution"
                                                    type="ResolutionType" maxOccurs="unbounded"/>
                                        <xs:element name="BrightnessIntensity">
                                            <xs:complexType>
                                                <xs:sequence>
                                                    <xs:element
                                                        name="MaximumValue"
                                                      type="xs:positiveInteger"/>
                                                    <xs:element
                                                        name="CurrentValue"
                                                      type="xs:positiveInteger"/>
                                                </xs:sequence>
                                            </xs:complexType>
                                        </xs:element>
                                        <xs:element name="ColorDepth" type="ColorBitDepthType"/>
                                        <xs:element name="RefreshRate">
                                            <xs:complexType>
                                                <xs:sequence>
                                                    <xs:element
                                                        name="MaximumValue"
                                                      type="xs:positiveInteger"/>
                                                    <xs:element
                                                        name="CurrentValue"
                                                      type="xs:positiveInteger"/>
                                                </xs:sequence>
                                            </xs:complexType>
                                        </xs:element>
```

```
                                        <xs:element name="Contrast">
                                            <xs:complexType>
                                                <xs:sequence>
                                                    <xs:element
                                                        name="MaximumValue"
                                                        type="xs:positiveInteger"/>
                                                    <xs:element
                                                        name="CurrentValue"
                                                        type="xs:positiveInteger"/>
                                                </xs:sequence>
                                            </xs:complexType>
                                        </xs:element>
                                        <xs:element name="Saturation">
                                            <xs:complexType>
                                                <xs:sequence>
                                                    <xs:element
                                                        name="MaximumValue"
                                                        type="xs:positiveInteger"/>
                                                    <xs:element
                                                        name="CurrentValue"
                                                        type="xs:positiveInteger"/>
                                                </xs:sequence>
                                            </xs:complexType>
                                        </xs:element>
                                    </xs:sequence>
                                </xs:complexType>
                            </xs:element>
                            <xs:element name="AudioOutput">
                                <xs:complexType>
                                    <xs:sequence>
                                        <xs:element name="Format" type="AudioFormat"/>
                                        <xs:element name="Power" type="xs:positiveInteger"/>
                                        <xs:element name="SignalNoiseRatio" type="xs:float"/>
                                    </xs:sequence>
                                </xs:complexType>
                            </xs:element>
                            <xs:element name="PowerConsumption">
                                <xs:complexType>
                                    <xs:sequence>
                                        <xs:element name="PowerSource">
                                            <xs:complexType>
                                                <xs:choice>
                                                    <xs:element
                                                        name="Battery"/>
                                                    <xs:element
                                                        name="ElectricityCable"/>
                                                </xs:choice>
                                            </xs:complexType>
                                        </xs:element>
                                        <xs:element name="BatteryCapacity" type="xs:positiveInteger"
                                                    minOccurs="0"/>
                                    </xs:sequence>
                                </xs:complexType>
                            </xs:element>
                            <xs:element name="Memory">
                                <xs:complexType>
                                    <xs:sequence>
                                        <xs:element name="Type" type="xs:string"/>
                                        <xs:element name="StorageCapacity" type="xs:string"/>
                                        <xs:element name="ReadPerformance">
                                            <xs:complexType>
                                                <xs:sequence>
                                                    <xs:element
                                                        name="Throuput"
                                                        type="xs:positiveInteger"/>
                                                    <xs:element
                                                        name="AccesTime"
                                                        type="xs:float"/>
                                                    <xs:element
                                                        name="CycleTime"
                                                        type="xs:float"/>
                                                </xs:sequence>
                                            </xs:complexType>
                                        </xs:element>
                                        <xs:element name="WritePerformance">
                                            <xs:complexType>
                                                <xs:sequence>
                                                    <xs:element
                                                        name="Throuput"
                                                        type="xs:positiveInteger"/>
                                                    <xs:element
                                                        name="AccesTime"
                                                        type="xs:float"/>
                                                    <xs:element
                                                        name="CycleTime"
                                                        type="xs:float"/>
```

```
                                                                                </xs:sequence>
                                                                        </xs:complexType>
                                                                </xs:element>
                                                        </xs:sequence>
                                                </xs:complexType>
                                        </xs:element>
                                        <xs:element name="CPU">
                                                <xs:complexType>
                                                        <xs:sequence>
                                                                <xs:element name="ProcessFrequency" type="xs:double"/>
                                                                <xs:element name="Cores" maxOccurs="unbounded">
                                                                        <xs:complexType>
                                                                                <xs:sequence>
                                                                                        <xs:element
                                                                                                name="Param1"/>
                                                                                        <xs:element
                                                                                                name="Param2"/>
                                                                                        <xs:element
                                                                                                name="Param3"/>
                                                                                </xs:sequence>
                                                                        </xs:complexType>
                                                                </xs:element>
                                                                <xs:element name="PowerConsumption" type="xs:float"/>
                                                                <xs:element name="Architecture">
                                                                        <xs:simpleType>
                                                                                <xs:restriction base="xs:string">
                                                                                        <xs:enumeration value="32
                                                                                                Bits"/>
                                                                                        <xs:enumeration value="64
                                                                                                Bits"/>
                                                                                        <xs:enumeration value="128
                                                                                                Bits"/>
                                                                                </xs:restriction>
                                                                        </xs:simpleType>
                                                                </xs:element>
                                                        </xs:sequence>
                                                </xs:complexType>
                                        </xs:element>
                                        <xs:element name="TrancodingCapabilities">
                                                <xs:complexType>
                                                        <xs:sequence>
                                                                <xs:element name="InputFormat" type="VideoFormat"
                                                                        maxOccurs="unbounded"/>
                                                                <xs:element name="OutputFormat" type="VideoFormat"
                                                                        maxOccurs="unbounded"/>
                                                                <xs:element name="TrancodingAverageSpeed"
                                                                        type="xs:unsignedLong"/>
                                                        </xs:sequence>
                                                </xs:complexType>
                                        </xs:element>
                                        <xs:element name="StorageCapacity" type="xs:unsignedLong"/>
                                        <xs:element name="Buffers">
                                        <xs:complexType>
                                                <xs:sequence>
                                                        <xs:element name="Type" type="xs:string"/>
                                                        <xs:element name="Size" type="xs:positiveInteger"/>
                                                </xs:sequence>
                                        </xs:complexType>
                                        </xs:element>
                                </xs:sequence>
                        </xs:complexType>
                </xs:element>
                <xs:complexType name="NetworkCardType">
                        <xs:sequence>
                                <xs:element name="Protocol" type="xs:string"/>
                                <xs:element name="BitRate" type="xs:positiveInteger"/>
                                <xs:element name="Frequency" type="xs:float"/>
                                <xs:element name="PowerConsumption" type="xs:float"/>
                        </xs:sequence>
                </xs:complexType>
</xs:schema>
```

2. XSD visual format





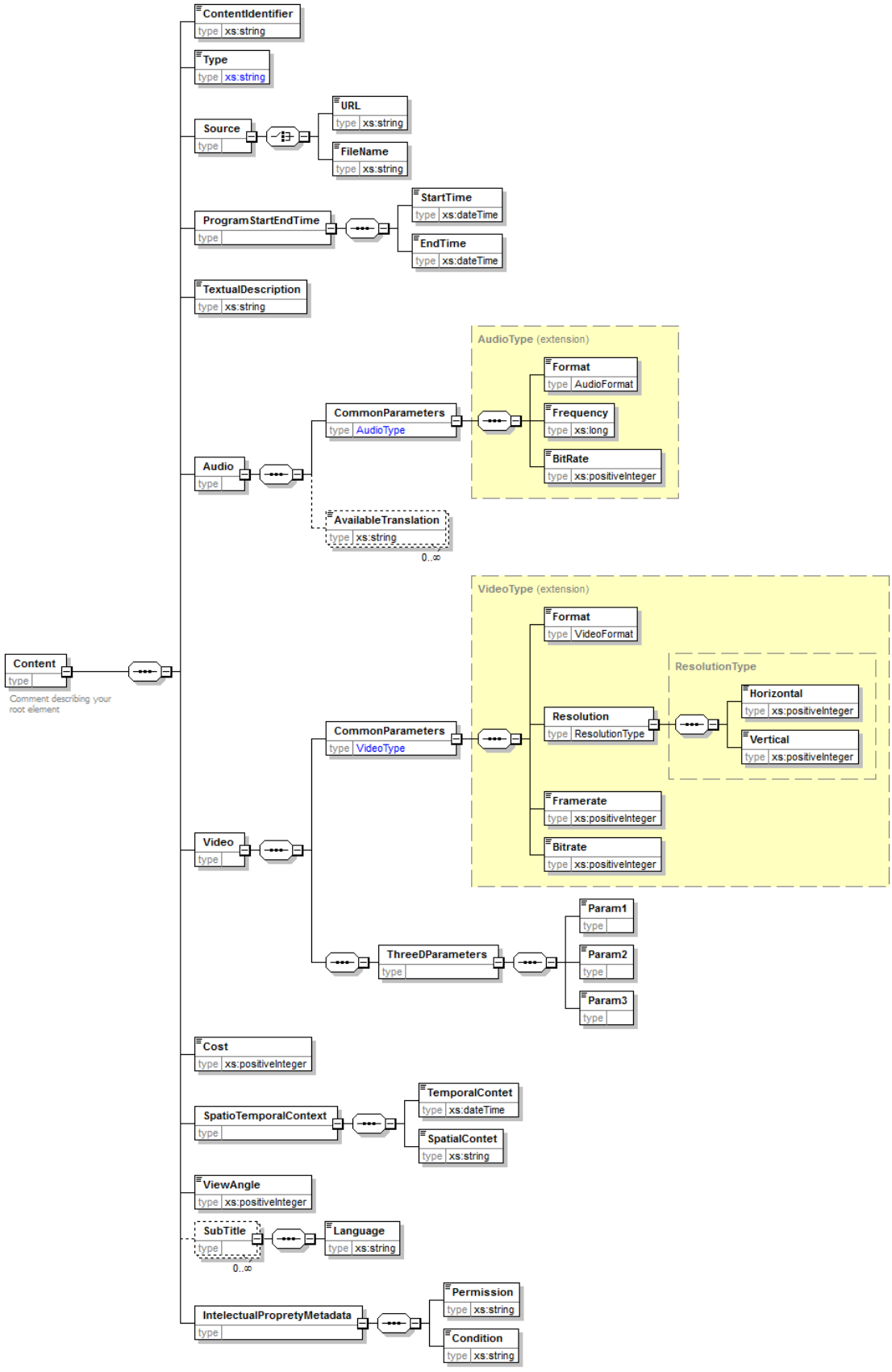
c) Content metadata

1. XSD code

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:ENVISION ="http://www.ENVISION-project.org/ENVISION"
elementFormDefault="qualified" attributeFormDefault="unqualified">
	<xs:include schemaLocation="ENVISIONTypesDeclaration.xsd"/>
	<xs:element name="Content">
		<xs:annotation>
			<xs:documentation>Comment describing your root element</xs:documentation>
		</xs:annotation>
		<xs:complexType>
			<xs:sequence>
				<xs:element name="ContentIdentifier" type="xs:string"/>
				<xs:element name="Type">
					<xs:simpleType>
						<xs:restriction base="xs:string">
							<xs:enumeration value="Audio"/>
							<xs:enumeration value="Video"/>
							<xs:enumeration value="Audio/Video"/>
							<xs:enumeration value="Html"/>
							<xs:enumeration value="Txt"/>
						</xs:restriction>
					</xs:simpleType>
				</xs:element>
				<xs:element name="Source">
					<xs:complexType>
						<xs:choice>
							<xs:element name="URL" type="xs:string"/>
```

```
                                                <xs:element name="FileName" type="xs:string"/>
                                        </xs:choice>
                                </xs:complexType>
                        </xs:element>
                        <xs:element name="ProgramStartEndTime">
                                <xs:complexType>
                                        <xs:sequence>
                                                <xs:element name="StartTime" type="xs:dateTime"/>
                                                <xs:element name="EndTime" type="xs:dateTime"/>
                                        </xs:sequence>
                                </xs:complexType>
                        </xs:element>
                        <xs:element name="TextualDescription" type="xs:string"/>
                        <xs:element name="Audio">
                                <xs:complexType>
                                        <xs:sequence>
                                                <xs:element name="CommonParameters">
                                                        <xs:complexType>
                                                                <xs:complexContent>
                                                                        <xs:extension
                                                                                base="AudioType"/>
                                                                </xs:complexContent>
                                                        </xs:complexType>
                                                </xs:element>
                                                <xs:element name="AvailableTranslation" type="xs:string"
                                                        minOccurs="0" maxOccurs="unbounded"/>
                                        </xs:sequence>
                                </xs:complexType>
                        </xs:element>
                        <xs:element name="Video">
                                <xs:complexType>
                                        <xs:sequence>
                                                <xs:element name="CommonParameters">
                                                        <xs:complexType>
                                                                <xs:complexContent>
                                                                        <xs:extension
                                                                                base="VideoType"/>
                                                                </xs:complexContent>
                                                        </xs:complexType>
                                                </xs:element>
                                                <xs:sequence>
                                                        <xs:element name="ThreeDParameters">
                                                                <xs:complexType>
                                                                        <xs:sequence>
                                                                                <xs:element
                                                                                name="Param1"/>
                                                                                <xs:element
                                                                                name="Param2"/>
                                                                                <xs:element
                                                                                name="Param3"/>
                                                                        </xs:sequence>
                                                                </xs:complexType>
                                                        </xs:element>
                                                </xs:sequence>
                                        </xs:sequence>
                                </xs:complexType>
                        </xs:element>
                        <xs:element name="Cost" type="xs:positiveInteger"/>
                        <xs:element name="SpatioTemporalContext">
                                <xs:complexType>
                                        <xs:sequence>
                                                <xs:element name="TemporalContet" type="xs:dateTime"/>
                                                <xs:element name="SpatialContet" type="xs:string"/>
                                        </xs:sequence>
                                </xs:complexType>
                        </xs:element>
                        <xs:element name="ViewAngle" type="xs:positiveInteger"/>
                        <xs:element name="SubTitle" minOccurs="0" maxOccurs="unbounded">
                                <xs:complexType>
                                        <xs:sequence>
                                                <xs:element name="Language" type="xs:string"/>
                                        </xs:sequence>
                                </xs:complexType>
                        </xs:element>
                        <xs:element name="IntelectualPropretyMetadata">
                                <xs:complexType>
                                        <xs:sequence>
                                                <xs:element name="Permission" type="xs:string"/>
                                                <xs:element name="Condition" type="xs:string"/>
                                        </xs:sequence>
                                </xs:complexType>
                        </xs:element>
                </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

2. XSD visual format

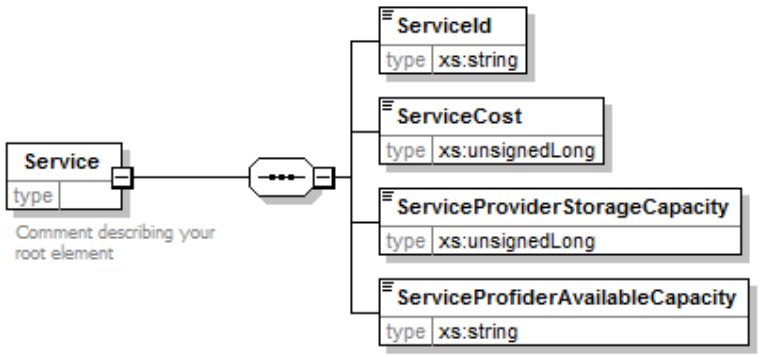


d) Service metadata

1. XSD code

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"  xmlns:ENVISION ="http://www.ENVISION-project.org/ENVISION"
elementFormDefault="qualified" attributeFormDefault="unqualified">
	<xs:include schemaLocation="ENVISIONTypesDeclaration.xsd"/>
	<xs:element name="Service">
		<xs:annotation>
			<xs:documentation>Comment describing your root element</xs:documentation>
		</xs:annotation>
		<xs:complexType>
			<xs:sequence>
				<xs:element name="ServiceId" type="xs:string"/>
				<xs:element name="ServiceCost" type="xs:unsignedLong"/>
				<xs:element name="ServiceProviderStorageCapacity" type="xs:unsignedLong"/>
				<xs:element name="ServiceProfiderAvailableCapacity" type="xs:string"/>
			</xs:sequence>
		</xs:complexType>
	</xs:element>
</xs:schema>
```

2. XSD visual format



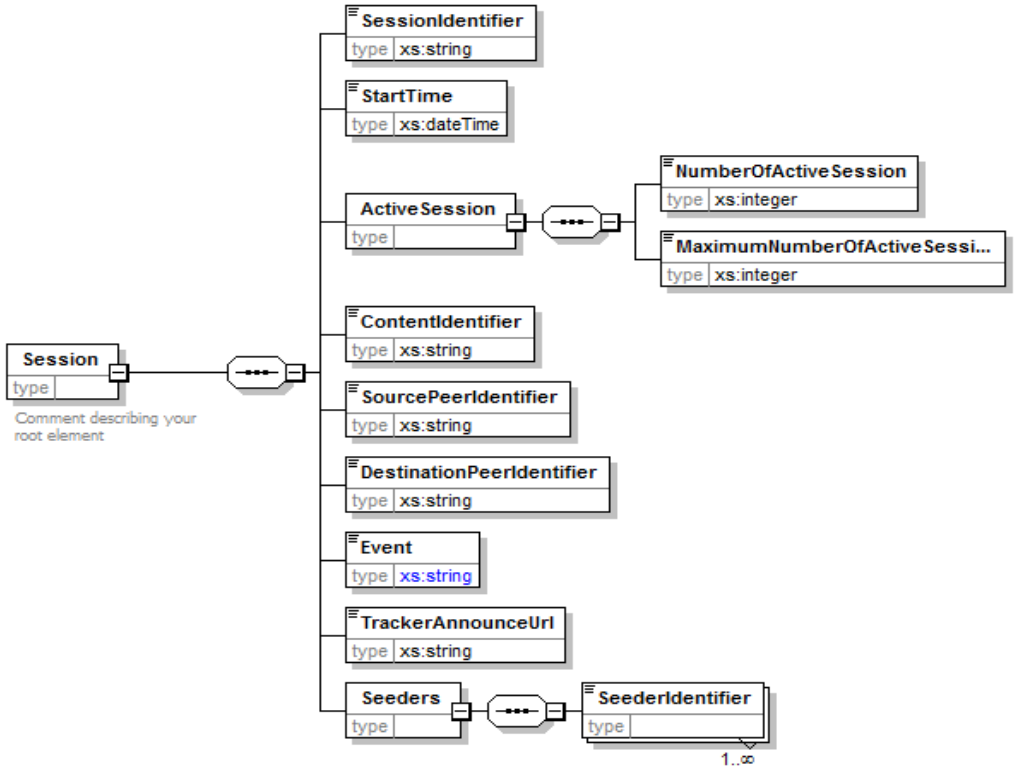
e) Session metadata

1. XSD code

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"  xmlns:ENVISION ="http://www.ENVISION-project.org/ENVISION"
elementFormDefault="qualified" attributeFormDefault="unqualified">
	<xs:element name="Session">
		<xs:annotation>
			<xs:documentation>Comment describing your root element</xs:documentation>
		</xs:annotation>
		<xs:complexType>
			<xs:sequence>
				<xs:element name="SessionIdentifier" type="xs:string"/>
				<xs:element name="StartTime" type="xs:dateTime"/>
				<xs:element name="ActiveSession">
					<xs:complexType>
						<xs:sequence>
							<xs:element name="NumberOfActiveSession"
type="xs:integer"/>
							<xs:element name="MaximumNumberOfActiveSession"
type="xs:integer"/>
						</xs:sequence>
					</xs:complexType>
				</xs:element>
				<xs:element name="ContentIdentifier" type="xs:string"/>
```

```
                                        <xs:element name="SourcePeerIdentifier" type="xs:string"/>
                                        <xs:element name="DestinationPeerIdentifier" type="xs:string"/>
                                        <xs:element name="Event">
                                                <xs:simpleType>
                                                        <xs:restriction base="xs:string">
                                                                <xs:enumeration value="Started"/>
                                                                <xs:enumeration value="Completed"/>
                                                                <xs:enumeration value="Stopped"/>
                                                        </xs:restriction>
                                                </xs:simpleType>
                                        </xs:element>
                                        <xs:element name="TrackerAnnounceUrl" type="xs:string"/>
                                        <xs:element name="Seeders">
                                                <xs:complexType>
                                                        <xs:sequence>
                                                                <xs:element name="SeederIdentifier"
maxOccurs="unbounded"/>
                                                        </xs:sequence>
                                                </xs:complexType>
                                        </xs:element>
                                </xs:sequence>
                        </xs:complexType>
                </xs:element>
</xs:schema>
```

2. XSD visual format



f) Peer metadata

1. XSD code

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:ENVISION ="http://www.ENVISION-project.org/ENVISION"
elementFormDefault="qualified" attributeFormDefault="unqualified">
	<xs:include schemaLocation="ENVISIONTypesDeclaration.xsd"/>
	<xs:element name="Peer">
		<xs:annotation>
			<xs:documentation>Comment describing your root element</xs:documentation>
		</xs:annotation>
		<xs:complexType>
			<xs:sequence>
				<xs:element name="Identifier" type="xs:string"/>
				<xs:element name="IpAddress" type="xs:string" maxOccurs="unbounded"/>
				<xs:element name="ListeningPort" type="xs:positiveInteger" maxOccurs="unbounded"/>
				<xs:element name="RequestedPieces">
					<xs:complexType>
						<xs:sequence>
							<xs:element name="PieceHash" minOccurs="0"
                                                                maxOccurs="unbounded">
                                                                <xs:simpleType>
                                                                        <xs:restriction base="xs:string">
                                                                                <xs:length value="20"/>
                                                                        </xs:restriction>
                                                                </xs:simpleType>
                                                        </xs:element>
                                                </xs:sequence>
                                        </xs:complexType>
                                </xs:element>
                                <xs:element name="AvailablePieces">
                                        <xs:complexType>
                                                <xs:sequence>
                                                        <xs:element name="PieceHash" minOccurs="0"
                                                                maxOccurs="unbounded">
                                                                <xs:simpleType>
                                                                        <xs:restriction base="xs:string">
                                                                                <xs:length value="20"/>
                                                                        </xs:restriction>
                                                                </xs:simpleType>
                                                        </xs:element>
                                                </xs:sequence>
                                        </xs:complexType>
                                </xs:element>
                                <xs:element name="State">
                                        <xs:simpleType>
                                                <xs:restriction base="xs:string">
                                                        <xs:enumeration value="Chocked"/>
                                                        <xs:enumeration value="Unchocked"/>
                                                        <xs:enumeration value="Interested"/>
                                                </xs:restriction>
                                        </xs:simpleType>
                                </xs:element>
                                <xs:element name="MaximumWantedSources " type="xs:positiveInteger"/>
                        </xs:sequence>
                </xs:complexType>
        </xs:element>
</xs:schema>
```

2. XSD visual format