DIOMEDES

SEVENTH FRAMEWORK PROGRAMME

DIstribution Of Multi-view Entertainment using content aware DElivery Systems

DIOMEDES

Grant Agreement Number: 247996

D2.3

Final reference system architecture report

Document description	
Name of document	Final reference system architecture report
Abstract	This document describes the final reference system architecture of DIOMEDES based on D2.1 and D2.2.
Document identifier	D2.3
Document class	Deliverable
Version	1.0
Author(s)	N. Just (IRT), M. Laabs (IRT), D. Driesnack (IRT)/ S. Dogan (UNIS), E. Ekmekcioglu (UNIS), H. Kodikara Arachchi (UNIS), S. Worrall (UNIS), A. Kondoz (UNIS), A. Fernando (UNIS)/ P. Aichroth (IDMT), A. Franck (IDMT), J. Hasselbach (IDMT), A. Haupt (IDMT), T. Korn (IDMT)/, C. Goktug Gurler (KOC)/ T. Adari (OPTEC)/ Z. Gaál (HOL), K. Lackner (HOL), P. Kovacs (HOL)/ E. Dogan (ARC), K. Aydoğdu (ARC), H. Gokmen (ARC)
QAT team	E. Ekmekcioglu (UNIS), T. Korn (IDMT)
Date of creation	09. November 2011
Date of last modification	31 January 2012
Status	Final Version
Destination	European Commission
WP number	WP2

TABLE OF CONTENTS

1 INTRODUCTION 6

2 SYSTEM ARCHITECTURE 7

2.1 Server Architecture 9
2.1.1 3D Content Server Module 9
2.1.2 DVB-T Transmission Module 9
2.1.3 TsChunker 10
2.1.4 Access Control & Content Registration Module (“Security Server”) 10
2.1.5 Main Seed Server Module 12
2.2 User Terminal Architecture 13
2.2.1 DVB-T Receiving Module 14
2.2.2 P2P Software Module 14
2.2.3 Authentication + Decryption Module (“Security Client”) 15
2.2.4 Control Module 15
2.2.5 Adaptation Decision Engine Module 16
2.2.6 Interaction Device 18
2.2.7 AV Synchronisation and Demux and Buffer Module 19
2.2.8 Audio Cluster 21
2.2.9 Video Cluster 23
2.3 Communication within the User Terminal 29
2.4 Transport Container 31
2.5 Synchronisation 31
2.5.1 Synchronisation of DVB and P2P 31
2.5.2 Synchronisation of audio and video clusters 31

3 P2P NETWORK ARCHITECTURE 32

3.1 Overview 32
3.2 Currently Available Solutions 32
3.3 Differences of the DIOMEDES P2P from currently available solutions 35
3.4 Adaptive streaming in the DIOMEDES P2P 36

4 RISK ASSESSMENT 38

5 CONCLUSIONS 40

APPENDIX A: GLOSSARY OF ABBREVIATIONS 42

LIST OF FIGURES

Figure 1 - Overall System Architecture 8

Figure 2 - Adaptation Decision Engine Module's functional overview 17

Figure 3 - Structure of the AV-Sync, Demux and Buffer Module 20

Figure 4 - Internal structure of the audio cluster 22

Figure 5 - Video cluster architecture consisting of a single PC for a single input display 24

Figure 6 - Video cluster architecture consisting of three PCs, where decoders are split to two PCs and the renderer is located in the third PC 24

Figure 7 - - Flow Diagram: Content consumption 30

Figure 8 - PSNR over Time (1 GOP ~= 0.5 sec) Wildlife 33

Figure 9 - Bitrate distribution over GOP 34

Figure 10 - The overhead of over provisioning in CBR 34

LIST OF TABLES

Table 1 – Security algorithms 11

Table 2 - P2P chunk header format as generated by the “Security Server” (taken from D5.4) 12

Table 3 – Adaptation requirements in DIOMEDES 18

Table 4 - Padding overhead for different chunk sizes 35

1 INTRODUCTION

1.1 Purpose of the document

This deliverable is the final deliverable for the DIOMEDES system architecture. It is based on D2.2 and on the progress in the other work packages. Already finalised aspects (e.g. use cases, system requirements, A/V requirements) can be found in D2.2 and are not repeated in D2.3.

1.2 Scope of the work

The specified final system architecture described in this document forms the basis of the products of other work packages in DIOMEDES. More detailed information about the final specification for the A/V processing can be found in WP3 and the corresponding deliverables e.g. D3.4 (Report on the QoE model and the audio and visual attention models) and D3.5 (Report on 3D Audio / Video rendering and content adaptation). Final A/V coding and P2P distribution system architecture is more described in more detail in WP4 deliverables e.g. D4.4 (Report on the developed audio and video codecs) or D4.5 (Report on performance of integrated MD-SMVD and P2P system). With all these definitions described in this document the proof of concept prototype have been developed (WP5), integrated and demonstrated (WP6). Additionally, initial performance results of the prototype algorithms can be found in D5.2.

1.3 Objectives

This deliverable contains the final system architecture for DIOMEDES, describing the final reference system architecture with all included modules, the synchronisation of the delivery channels (P2P and DVB) and their essence (audio and video).

1.4 Structure of the document

The final system architecture of DIOMEDES is described in Chapter 2 including all module descriptions of the server and the user terminal architecture. Chapter 3 describes the P2P network architecture that is adopted for DIOMEDES. In Chapter 4, a short risk assessment is carried out and Chapter 5 concludes the conclusion of the document.

2 SYSTEM ARCHITECTURE

For high adaptability and flexibility, the overall system is divided into modules forming a loosely coupled and partially distributed system.

Most modules are implemented as autonomous processes communicating over sockets.

This is a flexible approach that makes use of sockets for the exchange of compressed multimedia data and JSON-RPC (JavaScript Object Notation – Remote Procedure Call) for control data.

Within the User Terminal the overall compressed multimedia data flow is considered in PUSH-mode from DVB/P2P (Digital Video Broadcasting, Peer to Peer) up until A/V decoding.

Video and audio rendering are realised as cluster based systems as these tasks use complex algorithms that demand for high processing power. Thanks to the distributed architecture design, this is easily supported by the overall socket communication concept.

Overall, this concept allows easy switching of the rendering modules and makes integration and debugging easier than a monolithic solution. The overhead of the local socket communication is negligible compared to e.g. decoding or DVB-T demodulation. Implementation and testing of components can be done independent of other system elements because missing components can be replaced with standard-software supporting MPEG2-TS (e.g. VLC can stream, Demux and playback media contained in MPEG2-TS).

Figure 1 gives the final overview of the modules the system has and shows how they communicate. The system is divided into two main parts, the User Terminal and the Server.

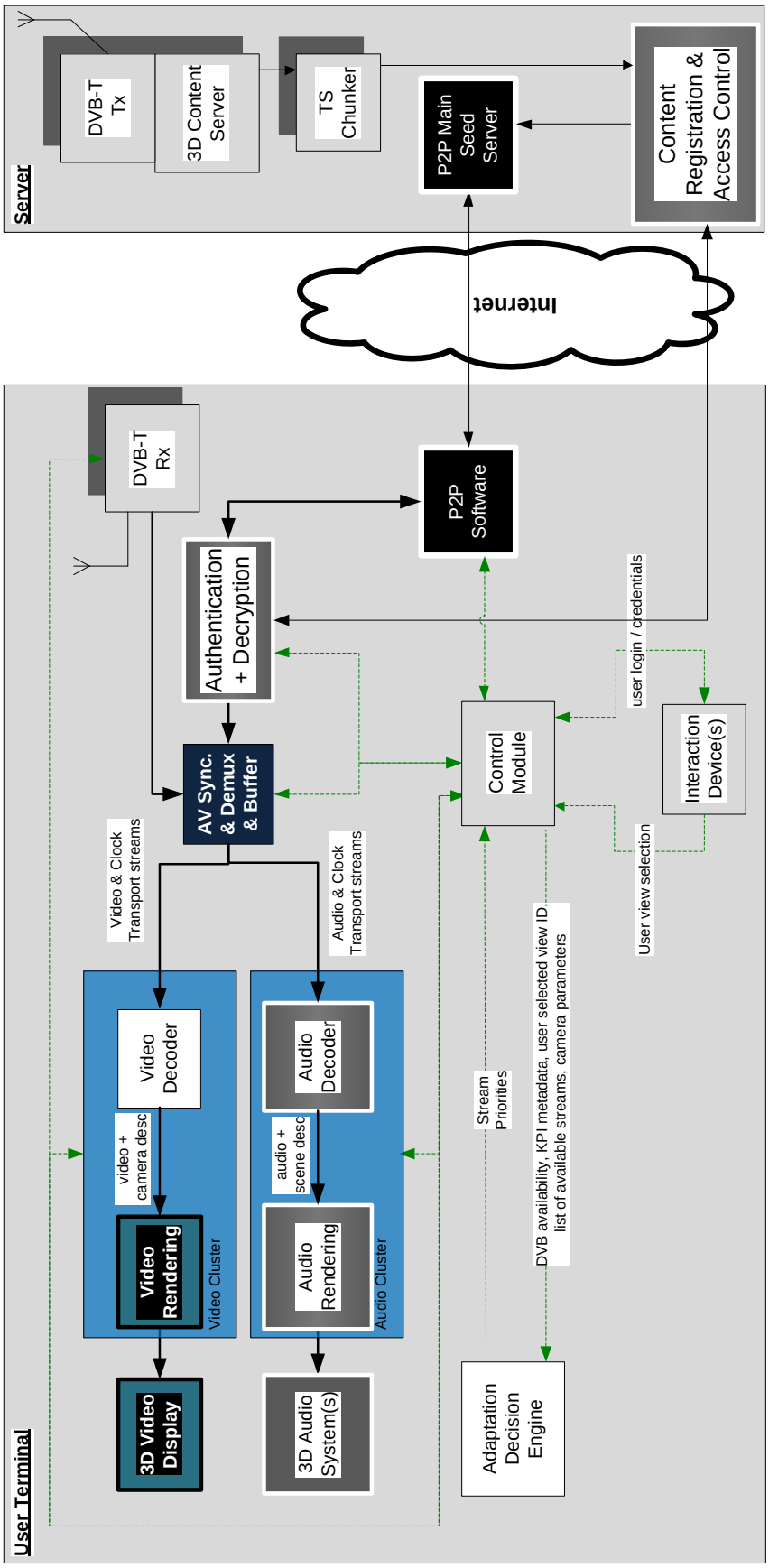

Figure 1 - Overall System Architecture

During the system architecture development, in order to speed up the development and to allow for working without specialized hardware (e.g. DVB-T, 3D display, P2P network, etc.) several simulation modules to cover such components have been implemented. For the P2P and DVB-Tx this is simply achieved by replacing them with MPEG2-TS streamers, for the A/V-decoding/rendering it is achieved by utilising the highly flexible VLC player.

The following sections give a detailed description of the modules depicted in Figure 1:

2.1 Server Architecture

The server's main tasks are DVB-T play-out and initial seeding of the P2P network. Related with those two main tasks the server is responsible for Content Registration & Access Control.

The 3D Content Server generates MPEG2-TS for DVB-T broadcast and transmission over the P2P Network. The P2P Main Seed Server feeds the content into the P2P Overlay Network.

A modern desktop hardware is used as a suitable platform for the server: Intel Core i7 Quad Processor, min. 8 GB DDR3 RAM, Solid State Disk, two Gigabit Ethernet Adapter. The operating system is Windows 7 (64bit).

2.1.1 3D Content Server Module

This module generates the broadcast services from available 3D Content. It multiplexes MPEG2-TS as input for the DVB-T Transmission Module and the P2P Main Seed Server Module. The TS is synchronously feed into the mentioned modules. For associating the content distributed via P2P with the content broadcasted via DVB-T, application signalling mechanisms standardized in ETSI TS 102 809 V1.1.1 (2010-01) is used. The URL to a document describing the corresponding P2P feed is coded by using this mechanism (e.g., http://www.diomedesWebServer.com/content/football.meta). Basically, for transmission through DVB two different types of video content can be created. It can be normal 2D or 3D stereoscopic content. In the first case it is possible to transmit the second view for generating stereoscopic 3D via P2P. In the second case P2P is used to provide additional views for generating multi-view 3D (see also D2.2, Use Cases).

Output:

- MPEG2-TS for processing in TsChunker
- MPEG2-TS for DVB-T Transmission Module

2.1.2 DVB-T Transmission Module

This module uses a DVB-T modulator and broadcasts the 3D stereo content received from the 3D Content Server Module.

The module could easily be replaced by other DVB modules e.g. DVB-T2, DVB-S2, DVB-C, DVB-IP or even other technologies if such distribution channels were preferred (Please note that this is not within the scope of this project). This module is working according to the relevant and available DVB standards. On top of these standards there are some modifications in DIOMEDES which are not standardised in DVB up to now. There are two

PIDs used for transmitting the stereoscopic video content over DVB. In DIOMEDES the lower number is used for the left view and accordingly the higher number for the right view.

Input:

- MPEG2-TS from 3D Content Server Module

Configuration Parameter:

- DVB-T modulator settings (see also D5.5)

2.1.3 TsChunker

TsChunker takes an MPEG TS multiplex as input (either from file or as stream) and provides blocks of data separated by PID (Program Identifier) and cut at GOP (Group of Pictures) borders as output. For audio there are both: a size and a time limit with the first chunk defining the amount of samples per chunk for all following chunks. The output is passed via SOAP (Service Oriented Architecture Protocol) to the Access Control & Content Registration Module, alongside with basic data about timing and view information that is read from a configuration file.

TsChunker also has basic analysis functionality in stream and file mode (can enable real-time speed limitation for that) to monitor presence of PIDs and display PCR (Program Clock Reference), PTS (Presentation Time Stamps) and time differences.

Input:

- MPEG TS (File/UDP)

Output (exclusive):

- SOAP (WSDL from Access Control & Content Registration Module)
- MPEG2 TS (UDP, Chunked, for test purposes only)

Configuration:

- Command line parameters (-h)
- Metadata JSON File defining PIDs, Views, Layers, etc.

2.1.4 Access Control & Content Registration Module (“Security Server”)

This module is implemented as a web service (including attachment support in streaming mode). The main purposes of this module are:

1. Access control:
    - to encrypt selected content in order to prevent unauthorized receivers from accessing content
    - to manage related key information (including key generation and storage), to prepare content for distribution via P2P network, i.e. generation of P2P chunks
2. Content registration:
    - to check content and mark (digitally sign) approved content in order to enable receivers to check content authenticity and prevent sharing of malware and unauthorized content

- support the upload of “user generated content” for further processing at the server side

3. Content discovery:
   - provide users with information about available P2P content (via “Security Client” and “Control Module” in the User Terminal)

The Security Server encrypts and digitally signs the transport stream chunks provided by the 3D Content Server. Moreover, this component is responsible for the management of keys and receivers - including key generation, storage and efficient key distribution using the available broadcast channels. The Security Server generates so called “chunk headers”, which store all information that is relevant for P2P distribution and security related tasks. The Security Server provides an API for registering (user generated) content, thus it serves as central entry point for new content.

Table 1 summarizes the algorithms for all security related tasks (please refer to D5.4 for detailed information on the depicted algorithms).

Task	**Proposed Algorithms/Schemes**
Content Encryption	AES (Rijndael), 128 bit key length, OFB mode
Cryptographic Hashes	SHA-256 (256 bit)
Signatures	RSA with SHA-256
Session Key Encryption	AES (Rijndael), 128 bit key length, OFB mode
Broadcast Encryption Scheme	Complete Subtree
Library	BouncyCastle (JAVA), SunJCE (JAVA)

Table 1 – Security algorithms

Input and Output

Input (see D5.4 for a detailed API description):

- A/V chunks generated by the TsChunker
- Content Metadata (in JSON-RPC file format)
- additional information required for P2P chunk generation, such as content ID, view ID, PID, PCR, metadata flag, media type flag, layer type (found in the header of chunks generated by TsChunker)
- Encoded media files (in case of “user generated content”)
- Requests for content information

Output (see D5.4 for a detailed API description):

- P2P chunks, which contain either A/V data, key information or content metadata
- Content information (for content discovery)

Byte	Parameter
1	Flags Bit 1 set: payload is encrypted Bit 2 set: payload contains enhancement layer information Bit 3 set: payload has been signed Bit 4 set: payload contains “content metadata” (instead of AV-content) Bit 5 set: payload contains audio data Bit 6/7: reserved for future use Bit 8 set: payload contains “key information”
2 - 17	Chunk ID: unique chunk identifier
18 - 33	Content ID: unique content identifier
34 - 37	PID: packet ID, used for identifying different elementary streams
38 - 43	PCR: program clock reference, used for synchronising P2P and DVB-T streams
44 - 47	Chunk count
48 - 303	Signature of the payload
304	View ID: identifies different video views
305 - 308	Payload size (in bytes)

Table 2 - P2P chunk header format as generated by the “Security Server” (taken from D5.4)

Table 2 describes the P2P chunk header format as it is generated by the Security Server. For detailed workflows regarding the “Security Server”, please refer to D5.4.

2.1.5 Main Seed Server Module

The main seed server is responsible for 5 key tasks:

The first and most important role of the main seed server is to initiate the distribution of the 3D content to a limited number of peers that reside inside the swarm. Once the content distribution is initiated, the activity of the main seed server starts to diminish as more chunks become available among the peers. The data flow from the main seed server to the peers is performed over the HTTP protocol. Thus a peer that cannot download the content from other peers within a time limit can always connect to the main seed server and receive the chunk. Therefore, the main seed server also works as a fall-back source in case P2P distribution fails to operate properly. However, the protocols that are adopted in the P2P distribution aim to minimize the server load.

The choice of HTTP is to enable interoperability with other HTTP based streaming solutions. Thus, peers can easily support HTTP streaming in server-client mode without any modification because in HTTP streaming the video is also split into segments and requested in a PULL fashion, similar to torrent-based solutions.

The second responsibility of the Main Seed Server is to behave as a tracker server for each session and help a new coming peer to find other peers in the swarm.

The third responsibility of the Main Seed Server is to store the metadata chunk that has to be forwarded to peers in order to initialize the receiving system. The metadata includes parameters that are critical for video rendering and P2P/DVB synchronisation such as camera calibration parameters and PID number of the streams. Therefore, in order to start a new session this chunk should be downloaded first.

Fourthly, the Main Seed Server stores the key information that is used for content authentication at the receiver side. Please note that the security scheme in the DIOMEDES project is much different than a standard torrent based system in which the chunks hash values are stored in the metadata file. In DIOMEDES however, the key is distributed to each peer in server-client manner in order to avoid security complexity in P2P network. Once the key has been received it is possible to authenticate the session chunks at the receiver side (see also 2.1.4 or D5.4 for a detailed description).

Finally, the Main Seed Server performs sanity checks on the content that is being uploaded. The following tests are performed in order to accept the content.

- The starting PCR value of each stream (PID) should be very close to each other.
- For each PID in the metadata file, there should be at least one chunk.
- The ending PCR value of each stream should be close to each other.
- The number of streams in the metadata chunk should be equal to the total number of PIDs.
- The GUID string of each content should be unique.

Input:

- Encrypted chunks
- Key for decryption
- Metadata

Output (exclusive):

- Encrypted chunks
- Key for decryption
- Metadata

2.2 User Terminal Architecture

As a platform for all modules except the Audio player modern desktop hardware is used: Intel Core i7 Quad Processor, min. 8 GB DDR3-1600, Solid State Disk, two Gigabit Ethernet Adapter. The operating system is Windows 7 (64bit). Video and Audio players are dedicated clusters and only for the audio rendering cluster 64bit Unix/Linux is used.

64bit OS is preferred over 32bit as the 3GB RAM limitation might be troublesome for working with multiple HD streams. Most software is running smoothly in 32bit compatibility mode, but it is needed that all required hardware drivers are available.

In addition to the already defined modules a Control Module is specified, which is responsible for high-level control. Said module knows all other modules with their respective configuration (e.g. IP, ports, etc.) and starts these (if on the same machine). It also takes control commands from the User Interface and sends DVB-Tuning commands and initialises P2P connections.

Furthermore, there is a log window that shows received and sent messages between modules. This makes easy to follow and debug module communications. And also, it can be used for being informed about the status of the modules.

2.2.1 DVB-T Receiving Module

This module receives the broadcast DVB-T signal and (after demodulating) transmits the MPEG2-TS to the AV Synchronisation and Demux and Buffer Module.

The module like its counterpart within the server could easily be replaced by other DVB modules.

Input:

- Channel Selection
- Control information

Output:

- MPEG2-TS received by DVB-T Rx (sent via UDP/IP)
- P2P content ID
- Status information

Configuration Parameter:

- DVB-T tuning parameters
- Destination modules' IP-addresses and port numbers for sending the TS

2.2.2 P2P Software Module

The user initiates a 3D content request through the "Control Module" (see Section 2.2.4), which notifies the P2P module about the GUID of the content. Following that, the P2P software establishes a connection with the Main Seed Server and receives both a metadata file and a key data file that is used by the security client at the receiver side. The metadata file includes information about the 3D content and the key information that is required by the security client to decrypt the received chunks.

Once the initialization period is over the Adaptation Decision Engine Module - ADEM (see Section. 2.2.5) sends the prioritization list of streams, which indicates their importance levels. Based on this information the P2P module initializes the downloading process and forwards any downloaded chunk (in sequential order) to the AV Synchronisation Module (see Section 2.2.7) via Authentication and Decryption Module (See Section 2.2.3). The AV Synchronisation module frequently updates the P2P module about the PCR of the DVB channel. Based on this information, the P2P module jumps to the correct chunk (if necessary) and maintains synchronised video streaming. For details of adaptation strategies and the architecture of the

P2P solution, please refer to D4.5. For the issues regarding the implementation, please refer to D5.3.

Input:

- Session GUID
- Metadata
- Stream prioritisation list from ADEM

Output:

- Encrypted chunks in correct order
- Key for decryption
- Metadata

2.2.3 Authentication + Decryption Module ("Security Client")

The purposes of this module are:

- to authenticate data coming from the P2P Software Module and provide feedback in case of failed authentication
- to decrypt data coming from the P2P Software Module
- to manage locally stored keys and broadcast key information
- to support the upload of user generated content, incl. user login required for upload request signing
- to support requesting content related information from the "Security Server"

Input (see D5.4 for a detailed API description):

- Encrypted P2P chunks
- Requests for content information
- Requests for content upload

Output:

- MPEG2-TS chunks (to AV Synchronisation Module)
- Status information regarding received P2P chunks (as the authentication result)

For detailed workflows regarding the "Security Client", please refer to D5.4.

2.2.4 Control Module

Control module is the centralized module that is aware of all other system modules. It forms the central control mechanism of the user terminal functions. Thus it knows about all other

modules and is able to control them by sending commands and receiving status information about the system.
All other modules communicate with control module via socket messages. For transmission of metadata and control data among modules, JSON-RPC (which is a remote procedure call protocol) is used.
The Control module is responsible for:

- Getting user login & credentials from interaction devices and sending this information to authentication & decryption module. The status of login (true/false) is sent back from Authentication & Decryption Module to Interaction Device via the Control Module.
- Sending status information about whole system to interaction devices. This information includes: DVB-T receiver status (signal information), audio/video clusters' status, alerts about low/high buffer levels (AV Synchronisation & Demux & Buffer Module), current bandwidth (P2P Software module).
- Triggering modules, e.g. start audio rendering, start/stop DVB-T receiving, initiate P2P download.
- Sending user preferences from interaction devices to system modules, e.g. display setup info, DVB-T channel selection, user requested viewpoint.
- Feeding the Adaptation Decision Engine Module with Key Performance Indicator (KPI) metadata and user requested viewpoint information.
- Getting P2P metadata, content description information from Content Registration & Access Control module and sending this information to related modules.
- Sending view priority list from Adaptation Decision Engine Module to P2P software module whenever ranking is changed.
- Feeding video & audio clusters with required viewpoint position which can be controlled by the user by the Interaction Module.

2.2.5 Adaptation Decision Engine Module

This module takes the following as inputs:

- Quality of Experience (QoE) estimated by the QoE model using the KPI metadata
- User requested view ID for view selection
- Availability of DVB streams
- List of available streams in the content server
- Camera parameters

The output of this module is:

- A list of media stream priorities for the P2P Software Module

The Adaptation Decision Engine Module's (ADEM) output is sent to the Control Module, which then relays the information to the P2P Software Module.

In a scenario where the P2P Software Module requests media streams whose bandwidth requirement exceeds the network capacity, the network may get congested. This may lead to random packet losses, which in turn may have a detrimental impact on the QoE of the

received media. In such a case, it is preferable that the P2P Software Module drops the streams that have the least impact on the perceived media quality (i.e., QoE) instead of random packet drops occurring due to congestion. The stream priority list generated by the ADEM is thus exploited by the P2P Software Module for deciding which stream(s) to be dropped.

The ADEM also responds to user interaction for interactive view selection. In this scenario, the function of the ADEM is to identify the best combination of available viewpoints for synthesising a user selected virtual viewpoint. In this way, the most suitable viewpoints are prioritised over the others so that the P2P Software Module retrieves viewpoints that are essential for synthesising the user selected virtual viewpoint accordingly. Figure 2 depicts the functional overview of the Adaptation Decision Engine Module.

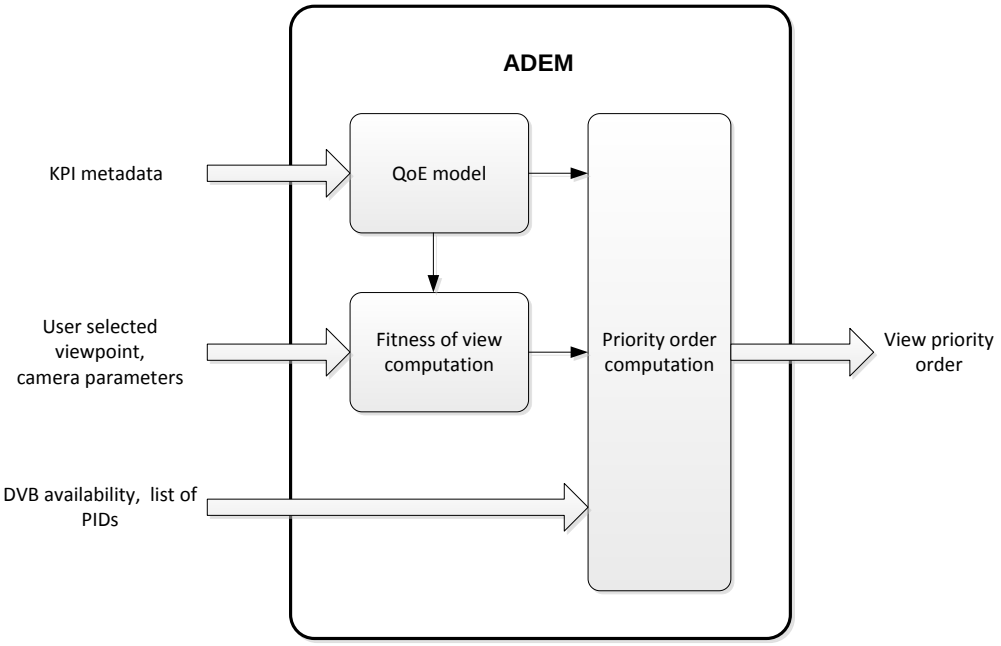

Figure 2 - Adaptation Decision Engine Module's functional overview

The adaptation of audio rendering is not controlled by the ADE. Audio rendering adaptation is controlled by

- audio rendering configuration (including information on display arrangement),
- audio scene input streams sent to the audio cluster and a
- viewpoint information sent to the audio cluster by the control module.

For the modular design of the audio cluster, see section 2.2.8 Audio cluster.

Fehler! Verweisquelle konnte nicht gefunden werden. summarises the adaptation requirements that are performed based on the corresponding adaptation inputs. It should be noted that the ADEM makes high level adaptation decisions while low level adaptation decisions are taken by the respective modules (i.e., Audio Cluster, Video Cluster and P2P Module), which perform the adaptation.

Adaptation Type	Adaptation Inputs	Adaptation Outcomes
Audio adaptation for audio stream type (source: DVB-T or P2P)	audio stream meta data: audio stream type	Audio input streams for rendering will be chosen by combining the available stream types and a stream priority scheme. Source of the audio stream to be decoded will be switched accordingly.
View selection	QoE, user selected viewpoint ID, display requirements, audio-visual data availability for delivery	from 4 views (+depths) to 2 views (stereo vision)
		from 4 views (+depths) to 1 view (+depth)
		from 4 views (+depths) to 1 view (mono vision)
		from 4 views (+depths) to user requested view(s)
Bandwidth (bit rate) adaptation	QoE, user selected viewpoint ID, display requirements, channel information	from High Quality to Low Quality using scalable layers (The exact adaptation approach is described in D3.5/D3.6.)

Table 3 – Adaptation requirements in DIOMEDES

2.2.6 Interaction Device

Interaction device is the interface of the system which a user can reach and communicate with the system components through this interface. It provides the user a graphical interface to start/stop modules, change module parameters dynamically, manage user preferences and select programs and program options. Based on the preferences, permissions, keys and choices, the corresponding content (e.g. 4-view video and 6-channel P2P audio/ 2-channel DVB audio) is processed and displayed.

- When the User Terminal is turned on, the Control Module and all the other modules in the User Terminal start in background. The user needs to get the content list from Access Control & Content Registration Module. Once the contents are listed on interaction device, the user can select desired content.
- Once these steps are completed, the user is able to operate the system and the Control Module sends “start streaming” commands to the DVB-T Receiving Module and the P2P Software Module. The user is able to switch between contents and select 2D/3D modes through the user interface.
- While the system is running, the Control Module sends system status information to the Interaction Device. Users are informed about the signal information of the DVB-T receiver, buffer status and current P2P bandwidth.
- User can upload any content to the Access Control & Content Registration Module. It is needed that the user logs to the system first in order to upload the desired content.

Provided authentication information (ID, and password) is passed to the Access Control & Content Registration Module via the Control Module.

- For the free-viewpoint viewing feature, the Interaction Device facilitates a user interface to acquire the user's selection of the desired viewpoint / viewing angle in the units the adaptation decision engine can interpret, based on the reference camera coordinate system.

2.2.7 AV Synchronisation and Demux and Buffer Module

The module is responsible for:

- Demultiplexing the transport stream arriving from the DVB-T and P2P paths in order to separate the buffers for video and audio.
- Buffering the audio streams and the video-layer streams (layers form the video-viewpoints).
- Holding several seconds of data (transport streams) before streaming it to the relevant cluster.
- Synchronising the audio and the video, the streams of the video views and the streams arriving from DVB-T and P2P.
- Sending the synchronised streams to the Video cluster & Audio cluster some frames before the indicated presentation time in order to leave the decoders enough time to decode and wait until it is time to send the frame for rendering.

Input:

- From P2P: multiple single-program transport streams. For each video view: base-layer stream, enhancement-layer stream, and depth-map stream. In addition, P2P delivers object based audio streams
- From DVB-T: single-program transport stream, containing: single or two video PIDs for single or stereoscopic view including audio PIDs and PCR information (PCR in a separate PID or PCR carried with the video).
- Configuration input: mode (DVB-T and P2P or P2P only), content-metadata including information about which PIDs exist, add stream, remove stream, set buffer size.

Output:

- Single-program transport stream with PID per layer (base / enhancement / depth-map) and the PCR PID, sent to the video cluster using UDP/IP.
- Single-program transport stream with PID per audio stream and the PCR PID sent to the audio cluster using UDP/IP.
- PCR packets sent to both clusters using UDP/IP Multicast (Optional – in order to compensate for a possible difference in the update-time of the Video and Audio clusters).
- Configuration output: alert about low and high buffer levels.

The PCRs, PTSs and DTSs originate from the same System Time Clock (STC) for all the streams. All the transport streams in the User Terminal are delivered using UDP/IP.

The following figure shows the high-level internal structure of the module:

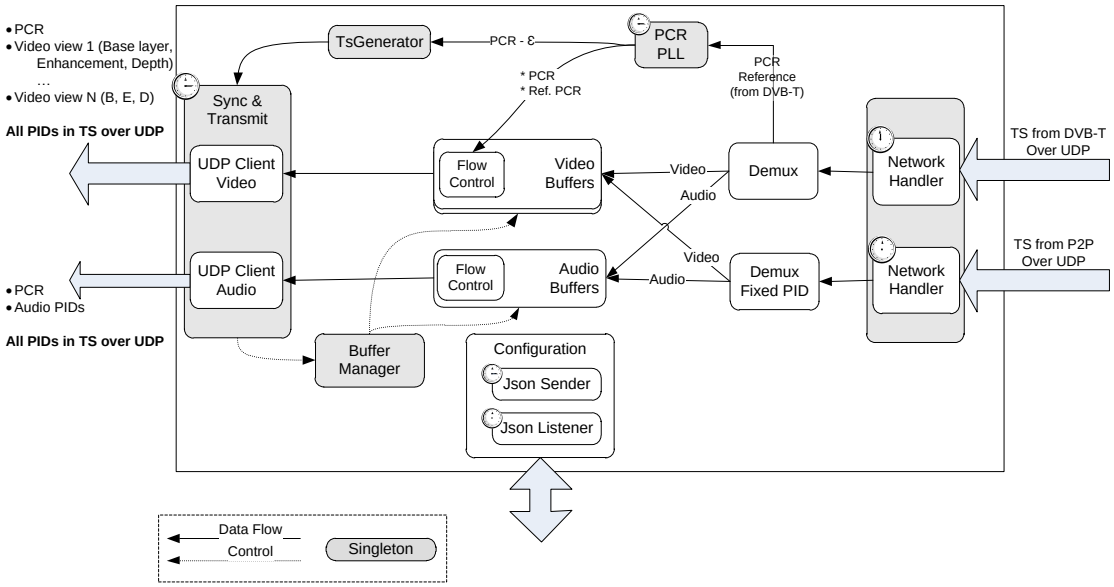

Figure 3 - Structure of the AV-Sync, Demux and Buffer Module

There are two scenarios regarding the PCR:

1. The stream from DVB-T is part of the program. The PCR can be extracted from this stream. It is transmitted with very low jitter and it can be used reliably to correct the internally generated PCR.
2. The program is composed only of the P2P streams (no DVB-T reception). The PCR received from the P2P network is not reliable since the network jitter is potentially large. In this case the internally generated PCR is used and transmitted to the relevant clusters.

Issues regarding buffer underrun/ overrun:

After a long time period the buffer can overrun or underrun because of PCR drifts between the original source and the internally generated PCR. This drift is rectified.

Preventing buffer underrun:

This is relevant only for the data arriving from P2P, since the DVB-T transmission speed is expected to be reliable.

When the buffer experiences a reduction in its filling level for some PIDs this might be due to a decrease in the network bandwidth. Before getting to a situation of buffer underrun, the P2P module should be informed about it (through the Control Module). It would then reduce the number of views (or streams within the views).

Preventing buffer overrun:

This is relevant only for the data arriving from P2P since the DVB-T transmission speed is expected to be reliable.

The buffer's filling level will increase when the Content Authentication Module sends the transport stream packets faster than the stream's actual bit-rate. This scenario can happen only if there is a drift in the flow control of the peer. The remedy is to send the P2P module periodically how much free space exists in the buffer, and then the P2P module would regulate the amount of chunks scheduled accordingly.

2.2.8 Audio Cluster

The software modules of the audio cluster conduct all audio processing steps to render audio content delivered over the P2P and the DVB transmission channels. The audio rendering is suitable to control 2D and 3D loudspeaker systems. A broad range of loudspeaker placements is supported. Audio content consists of a number of separated audio objects that are placed in 2D or 3D scenes with individual sound reproduction properties. This object oriented approach is used in state-of-the-art Wave field synthesis (WFS) reproduction systems and is also used here to drive the WFS loudspeaker systems or loudspeaker systems with a lower number of speakers.

The audio cluster of the receiver terminal is able to support the adaptation to different kinds of reproduction systems and audio processing approaches due to its modular design. The following main types of adaptation are considered here:

- Adaptation to different loudspeaker setups by choosing the audio rendering approach (e.g. WFS, lower resolution audio rendering) during the system configuration
- Adaptation to different scene reproduction approaches that allow audio rendering in conjunction with video reproduction on a defined display setup and listener/viewpoint setup
- Adaptation to different scene viewpoints chosen by the user
- Adaptation to different transmission paths and stream types (formats)

The audio cluster offers a JSON-RPC based control interface to the control module for viewpoint adaptation. Dependencies between modules are reduced to maintain a high inter-changeability. The following set of data fields is implemented to control viewpoint adaptation:

- Control of viewpoint position in relation to the video scene
- Control of viewpoint direction in relation to the video scene

The JSON messages are transferred via a TCP connection.

The audio cluster receives a transport stream over a UDP connection. The audio stream to be decoded is selected according to the incoming audio stream type (stream arriving through the P2P has higher priority). Audio latency time is determined by the timing data contained in the incoming audio streams. It can additionally be configured manually to adjust audio and video synchronicity.

The UDP-based audio scene transmission interface receives audio transport streams and object description data (embedded in the audio transport streams) from the AV Synchronisation & Demux & Buffer module. The audio scene transmission format enables synchronisation of audio and video cluster by applying clocking approaches described in MPEG-2 Systems standard that are also used in current DVB transmission. The transmission format has the following layered structure (starting from the highest layer, carrying the essence data):

- Audio scene essence: transmission of interleaved coded audio and scene description data (based on an MP4-container)
- above data, encapsulated within a Packetized Elementary Stream (PES), containing additional sync data like PTS (Presentation Time Stamps)
- above data, encapsulated within an MPEG2 - Transport Stream (MPEG2-TS), containing additional sync data like PCR (Program Clock Reference)

This structure offers PCR transmission to set the audio cluster's system time and adjust the audio resampling ratio. The encapsulated PES offers Presentation Time Stamps that allow a temporally defined play-out of the audio scene relative to the PCR time base.

The following Figure 4 shows the internal structure of the audio cluster and the I/O interfaces.

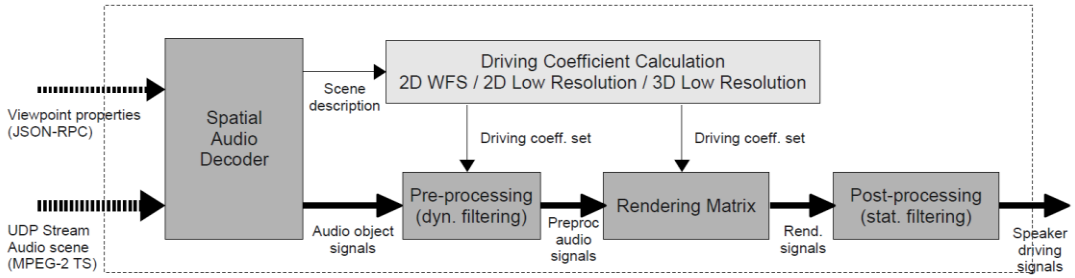

Figure 4 - Internal structure of the audio cluster

The implementation of the audio cluster for the demonstrator offers real time processing using one standard multi-CPU PC. Due to the modular structure of the audio cluster software, audio processing tasks are parallelized using multi-threading structures.

Input:

- Audio scene input encapsulated within an MPEG2-TS format - from AV Synchronisation & Demux & Buffer Module)
- JSON-RPC messages setting the user requested viewpoint parameters – from Control Module

Output:

- Audio signals for loudspeaker and subwoofer system

2.2.8.1 Audio Decoder

The audio decoder implements the following tasks:

- Accepting MPEG-2 TS audio scene data stream from UDP socket
- Unpacking the data stream by creating a set of coded audio data and a stream of scene description data
- Generating a set of decoded audio streams (support object based audio scene formats and channel based audio formats like 5.1 or 2.0 AC-3), using timing information for clock recovery and play-out synchronisation

- Buffering, re-sampling and outputting audio streams to the audio cluster rendering modules
- Converting scene description from the transmitted scene description format to internal scene description format and outputting the description data messages to the audio cluster rendering modules

A maximum number of 32 simultaneous audio objects are defined to work within DIOMEDES. MPEG-4/AAC is used for coding each audio object signal individually. Both variable and constant bitrate coding can be applied for audio scene coding.

The audio scene description is represented by a format of flat hierarchy (no grouping of audio objects and special treatment of groups). All hierarchical treatment of audio objects should be reserved for broadcast side and post production phase.

The audio scene description format counts for the variable reproduction scenarios by offering data fields that support the automatic scene adaptation to different display and listening setups.

2.2.8.2 Audio Rendering and Adaptation

The audio rendering modules of the audio cluster process the object based audio scene description and determine a set of driving coefficients adapted to the configured loudspeaker setup. These coefficient matrices control the processing of the audio object signals mainly in a parametric scale and delay matrix and in a time-variant pre-filtering stage. These processing steps lead to the loudspeaker driving signals. The design of the driving coefficient calculation algorithms is a crucial part of audio rendering. The different implemented audio rendering approaches differ mainly in the driving coefficient calculation.

Within processing of the audio scene description and coefficient calculation components, adaptation of the audio scene is implemented. The final audio rendering system and the implemented adaptation approaches are described in DIOMEDES Deliverable D3.5/D3.6.

2.2.9 Video Cluster

The video cluster is flexible in terms of architecture and can consist of up to 2 PC's devoted for scalable video decoding and one PC to carry out multi-view plus depth (MVD) rendering. Transport stream demultiplexer and video cluster controller modules can be located in one of the decoder PCs. The video controller is in charge of connecting the video cluster module to the main control module and hence the adaptation decision engine (ADE) module. It is capable of interrupting/ suspending the decoding process. The video cluster architecture can also be scaled down to one PC, especially for stereoscopic video player use case, where all modules (e.g. demultiplexer, controller, two scalable video decoders and stereoscopic video renderer) can be located in one PC. The single PC architecture is depicted in Figure 5, where the number of total video decoders is shown as N. Different supported display types (see section 2.2.9.7) incur a different number of decoder instances to be run in parallel. In the single PC architecture, raw (decoded) frame transport between the decoders and the renderer is done over a shared memory communication (can alternatively be done over TCP communication). Video decoder – video renderer decoded frame transmission process is explained in detail in Section 2.2.9.5. When the decoders are split into multiple PCs and rendering is done in a separate PC than the decoding operations and raw frames are transmitted over parallel InfiniBand connections using TCP communication protocol (this can be replaced with more efficient protocols to significantly increase the throughput, such as

RDMA). Because, shared memory access is applicable if the video decoders and the video renderer are physically located in the same PC. Figure 6 depicts this case. Some displays may necessitate more than one display input to be present at the same time. This affects the number of video renderer instances that need to be present, which feed separate display input ports. Some displays have the appropriate video rendering clusters integrated within them, which necessitate video stream communication over multiple network connections (e.g. Ethernet or InfiniBand connection for Holovisio display) from the user terminal PC.

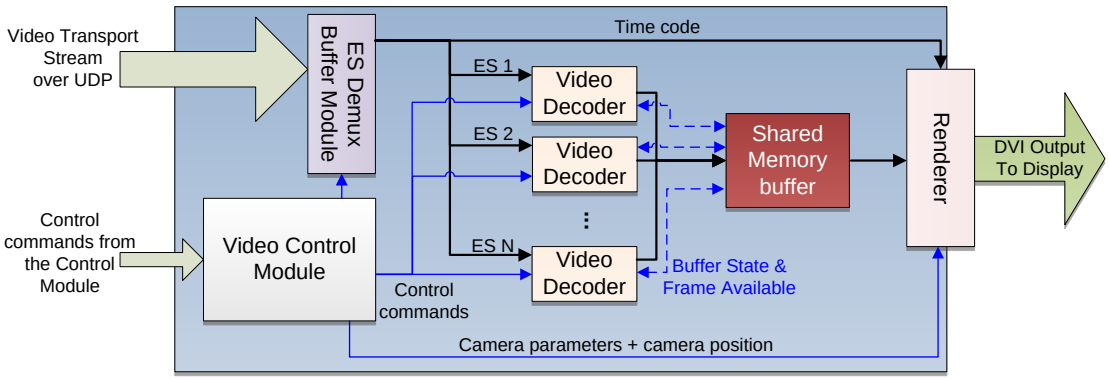

Figure 5 - Video cluster architecture consisting of a single PC for a single input display

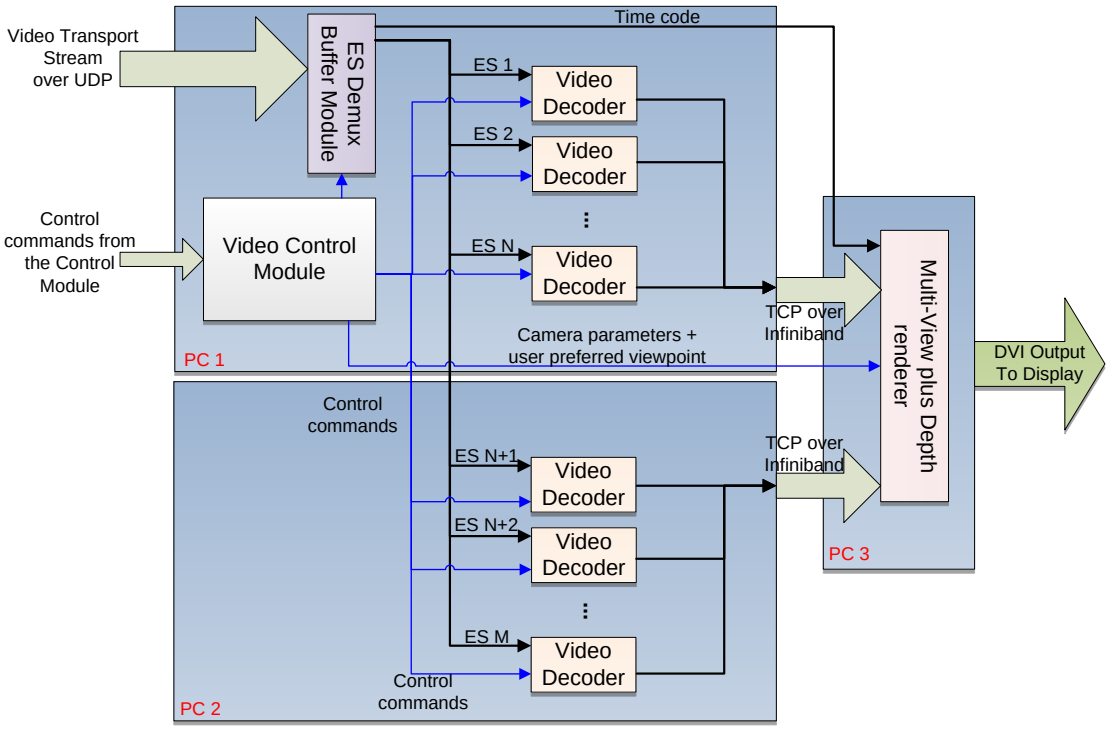

Figure 6 - Video cluster architecture consisting of three PCs, where decoders are split to two PCs and the renderer is located in the third PC

2.2.9.1 Video Controller

This module is responsible for controlling the operation of the rest of the modules in the video cluster. This module is in communication with the main control module, through which it

receives a set of parameters (often referred to as metadata) that are parsed and turned into a number of command messages sent to other modules. These include:

- The initialisation command sent to ES Demux: total number of PIDs, PIDs that need to be demultiplexed, PIDs' types and associations with each other, output UDP port numbers to direct the elementary streams to
- Commands sent to video decoders: start message (to let video decoders start decoding process), reset message (to let video decoders reset decoding process by deleting all internal buffers), stop message (to let video decoders stop the overall decoding process and become idle)
    - The video decoders are also informed at the beginning by the video controller about the input UDP port numbers they should expect the video streams from.
    - Video decoders listen to video controller command messages through pre-set and static TCP ports.
- Commands sent to the video renderer: three sets of different messages are sent to the video renderer:
    - Input channel configuration: total number of input media channels (e.g. camera views), corresponding intrinsic and extrinsic parameters of input camera views
    - Rendering mode (i.e. whether 2D, stereo, or multi-view)
    - User preferred viewpoint (to let the renderer synthesize the corresponding viewpoint and display)

2.2.9.2 Elementary Stream Demultiplexer (ES Demux)

This module is the entry point of encoded media in the video cluster. The synchronised transport stream multiplex, delivered through the AV synchronisation buffer module, is demultiplexed into individual H.264/SVC elementary streams. Each transport stream with a separate PID is demultiplexed and the corresponding NAL (Network Abstraction Layer) unit packets are stored temporarily in buffers before passing them to the video decoders. Since the SVC streams consist of two video coding layers (i.e. base quality layer and enhancement quality layer), these two layers need to be aggregated and sent to the video decoders together. However, they are carried separately in different transport streams. Hence, the internal buffer of the ES Demux handles aggregating the base layer and enhancement layer NAL units with the same presentation time (denoted by the Presentation Time Stamp - PTS) before sending them to the corresponding video decoder. Each SVC stream corresponding to a different camera viewpoint, or its depth map, is sent to a different video decoder over a separate UDP connection.

ES Demux receives the incoming transport stream multiplex over a UDP connection from AV synchronisation buffer module. The video controller initiates the ES Demux module by sending

- it the total number of PIDs it should de-multiplex,
- -the list of PIDs and their types (i.e. base, enhancement or depth),
- PIDs' associations with each other, and
- the output UDP port numbers to send the de-multiplexed elementary streams (each camera view or depth map stream is sent to its corresponding video decoder).

ES Demux also forwards the delivered PCR (i.e. clock, that is denoted by “Time code” in Figure 5) information after removing the transport stream header. Using the PCR information, the reference time is reconstructed in the audio and video renderer modules. This reconstruction is used to present the audio and video frames at the correct time and in synchronisation. Audio and video clusters receive the same PCR information delivered through AV synchronisation buffer module.

2.2.9.3 Video Decoder

This module receives the elementary streams for multi-view colour and multi-view depth videos from the ES Demux. The video decoder module is designed to run multiple instances (with no communication between them), while each of them is responsible for decoding the multi-layer elementary stream corresponding to one camera viewpoint, or depth map. This module is based on the Open SVC decoder software library (an optimised version of the SVC reference software developed by JVT for real-time decoding and that is governed by a Lesser General Public Licence) and is implemented in Windows OS. The detailed functionality of the video decoder unit is outlined in D4.3 - Interim report on the developed audio and video codecs, whereas the differences from the interim structure are outlined in D4.4 - Report on the developed audio and video codecs.

The decoder module receives the set of initial commands from the Video Controller in JSON message (sent via a TCP connection), which includes instructions about input UDP port numbers and the connection method with the renderer (e.g. shared memory, or TCP). Subsequently, multiple decoder instances are initiated. The Network Abstraction Layer (NAL) units corresponding to one camera viewpoint are delivered over a UDP connection to one of the decoder instances. Since the media packets are ordered in the previous synchronisation buffer, the decoders don’t handle a process of packet re-ordering. The receiver thread within the decoder is capable of aggregating the incoming stream for buffering. Consequently, NAL units are parsed to get parameter sets and entropy coded video information in the main thread. Base layer packets and Enhancement layer packets are decoded consecutively, meaning that base layer video can be decoded regardless of enhancement layer’s existence. If the NAL units of a base layer and the corresponding enhancement layer are delivered in the same UDP packet, the decoder decodes both, whereas if only the NAL unit of the base layer is delivered, the decoder decodes on the base layer frame. Decoded frames of each view/ depth are stored within each decoder instance’s own output frame buffer. Pictures are copied to another shared memory location to exchange raw frames with the renderer (refer to 2.2.9.5 for more details on decoder-renderer communication). A decoded frame is not deleted from the output picture buffer, until an Intra-coded (I) frame arrives to the decoder (forcing to flush decoded picture buffer).

There is a buffer of NAL units at the entrance of each decoder instance (i.e. a certain time delay between the receiving and main threads) to enable lost frame (i.e. lost GOP in the context of DIOMEDES P2P architecture) concealment because the decoders should track the picture-order-count continuity before the actual decoding process in order to trigger the necessary concealment procedure. At the same time, the main thread can be divided into sub-threads to handle parallel decoding at different hierarchies using Intel Thread Building Blocks library (e.g. either in macroblock level or GOP level, by exploiting the multi-processing core architecture more efficiently).

The video decoder is also responsible for extracting the Key Performance Indicator (KPI) messages from the elementary bit-stream (more specifically after the NAL units carrying the

parameter sets for decoding) and forwarding them to the Video Controller in a JSON-RPC message through TCP connection.

2.2.9.4 Video Renderer

The video renderer receives uncompressed views and depth maps from the video decoder(s), along with camera metadata and output configuration from the decoder control module and PCR clock from the ES Demux module.

The views are received as bitmaps in YUV colour space, while depth maps are read from the Y component of the depth stream. Camera metadata for the incoming views contains intrinsic and extrinsic matrices. The output is specified as number of views, the way these views are combined before presenting them to the attached 3D display (side-by-side or line interleaved in for stereoscopic displays, while custom patterns are specified individually for different multiview display models), as well as the position and resolution of the attached display (in a multi-monitor configuration). The output configuration also contains the central viewpoint selected by the user, from which the position of the other viewpoints (on both sides) are calculated based on content metadata and the capabilities of the attached display.

The decoder always supplies image and depth data with PTS values attached, and the renderer buffers them until they need to be presented according to the actual PCR. The renderer maintains an internal PCR clock, which is used to change the frame that's rendered. While the internal clock is precisely enough to maintain audio-video sync, it is synchronised with every PCR received from ES Demux. Frames with a PTS older than the actual PCR are freed from memory. Synchronisation of the presentation of frames with the PCR ensures strict audio-video synchrony.

The video renderer generates N views in real-time, according to the display attached, and mixes them in the format appropriate for the display. The generated 3D image is then sent via one or more DVI/HDMI connections to the display.

The video renderer algorithm can be implemented in parallel for each (or each group of) pixel(s). Thus the core algorithm is implemented on modern GPUs using geometry, vertex and fragment shaders. The target GPU is an NVDIA GTX 285 (minimum).

As the original quality of the incoming video streams might alter during rendering, pre-buffering of the stream is required leave time for post-processing to improve the quality. The renderer takes measures to reduce the "jumping" effect between the altered qualities (e.g. smoothly moving the virtual viewpoint to a position which is not affected by the loss of a view, which will soon occur)

2.2.9.5 Video Decoder – Video Renderer decoded frame communication

Shared storage or TCP (either over Ethernet or InfiniBand IPoIB) is the communication channel between the multiple decoders and the renderer. Multiple decoder processes that run concurrently have write access to a particular raw memory space and the renderer process has access to this memory space to read the uncompressed camera frames. As a shared resource, some means of race condition handling is also implemented to avoid memory flaws.

Depending on the decoder module's implementation, this module might work as a simple FIFO (First In First Out) stream where the decoders copy each decoded frame from its decoded picture buffer into this shared memory and the renderer takes them out. The reason why the decoded pictures are copied from the decoded picture buffers to the shared memory is that the decoded frames are deployed by the decoder for the motion compensation process of the subsequent frames to be decoded in the corresponding GOP. In case of networked operation over TCP, a simple protocol for implementing the same functionality has been implemented.

2.2.9.6 Video Decoder – Video Renderer frame synchronisation

The decoders add the Presentation Time Stamp (PTS) information taken from the header of the incoming UDP packets to the header of the raw uncompressed picture, while writing it to the shared storage buffer (please refer to D5.1 for more detail on the raw picture format). This way, the renderer module ensures that the decoded pictures are displayed in order and in synchronisation with the audio. The decoder module waits until the arrival of the next compressed access unit information. During this time, the renderer sleeps while sufficient amount of decoded frames are ready to be rendered in its own storage. At that point, the frames are not yet ordered by the PTS. The renderer renders a 3D image to be displayed using the decoded frames in its own storage. The renderer not only starts the rendering of the new 3D image, but also registers its PTS in the reconstructed timer. The time information (PCR) delivered to the renderer module goes to the internal timer to make the timer reconstruct the system's time more reliably and have a better synchronisation with the rest of the system. The time information as well as the elementary stream (ES) data is distributed via UDP over local host from the ES Demux. The role of the reconstructed timer is to trigger the renderer for the next registered PTS at the correct time and let the renderer push the rendered 3D image into the display. As the actual presentation ordering of the frames takes place in the reconstructed time module, the renderer can be informed about late frames as well (e.g. during registering a new PTS, it can be detected that the frame's presentation time has passed).

2.2.9.7 3D Video Display

The following 3D displays are and will be investigated until the end of the project:

- *Lightfield Display*:

  Input format: Holographics Lightfield

  Field Of View: 20-60

  Continuous horizontal parallax (auto-stereoscopy)

  Video renderer PC cluster included

- *JVC GD-463D10 46" 3D Display*

  Input format: Stereo video (left and right view), either side-by-side, or left and right views interlaced on alternate lines

  Xpol® circular polarizing method (passive stereo)

- *BEKO 26" multiview auto-stereoscopic TV prototype*

  Field of view: 5 sweet spots

  Native resolution: 1920x1200

  Inputs: VGA, HDMI

- *NewSight 42" Multiview AD3*

  Field of View: 8 views

  Native resolution: 1920x1200

  Inputs: VGA, HDMI

- *Panasonic TX-P50VT20E, 3D Display*

  Input format: Stereo video (left and right view), either side-by-side, top-and-bottom or frame-packing

  Active shutter glasses

  HDMI 1.4a compatible

  Native Resolution: 1920x1080

It is assumed that the exact input formats are known for the listed displays as it is a precondition for the renderer.

The physical interface between the Multi-view Display and the PC on which the video renderer module is located depends on the type of display.

For a single HDMI/DVI input display the video renderer PC is directly connected to the display via a free DVI/HDMI connection of the GPU.

For the Lightfield Display, where the rendering cluster is a part of the display, the physical interface is the Ethernet / InfiniBand network. The decoded raw video stream is transferred from the decoder to the head node of the render cluster, while the head node multicast the images to the rendering nodes, where they are rendered into light-field format, and the appropriate DVI/HDMI signals are output to the optical modules.

2.3 Communication within the User Terminal

In Figure 7 the flow chart of the summarised system communication on the User Terminal side is shown.

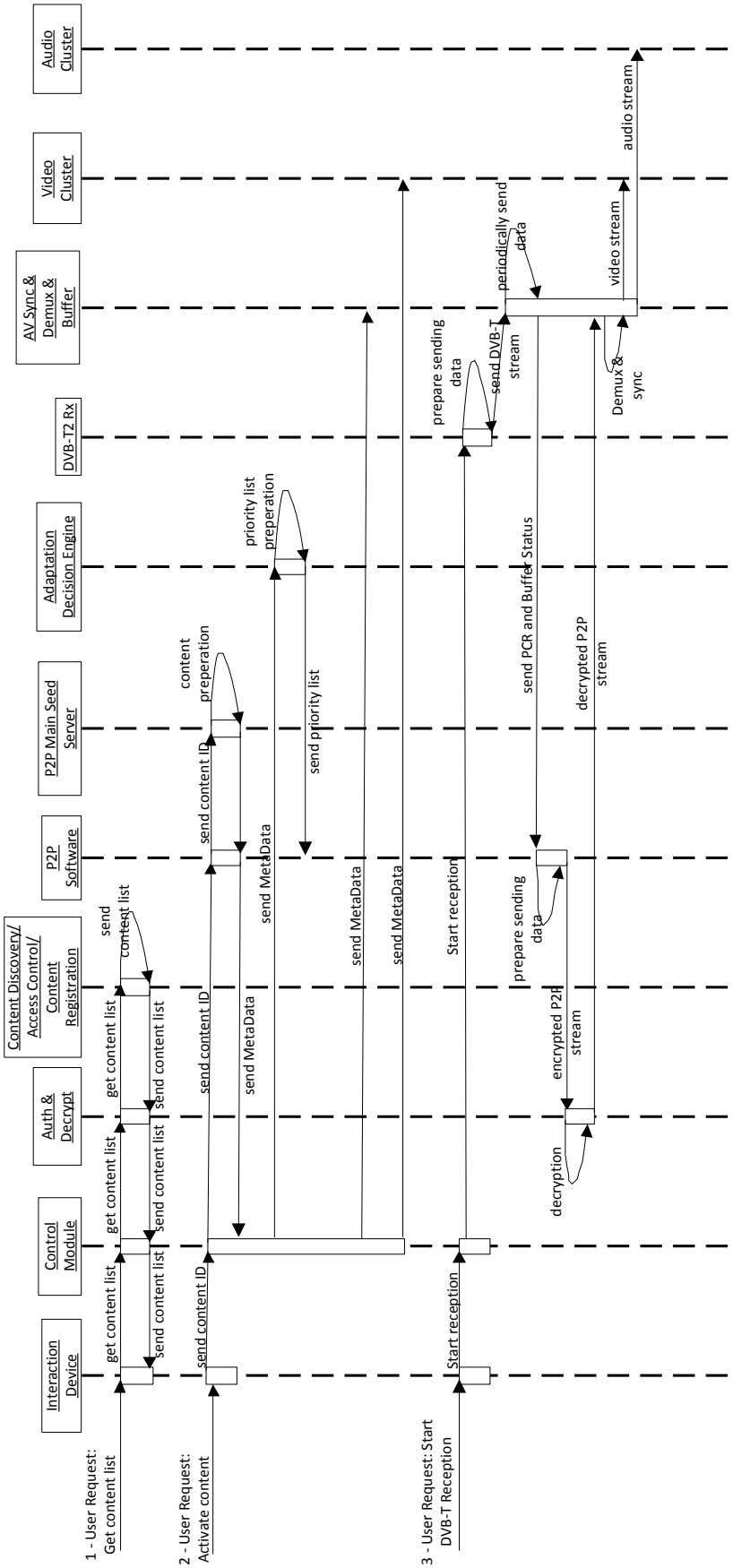

Figure 7 - - Flow Diagram: Content consumption

2.4 Transport Container

As it is mandatory in DVB the MPEG2-TS (ISO/IEC 13818-1) including a PCR is used as the media container. It is assumed that for P2P, the payload is transparent and thus the P2P Main Seed Server is fed with a TS stream from the DVB Play-out Server in parallel to the DVB-T play-out (see Server Architecture diagram in Figure 1). The mapping of different scalable layers, different views, and descriptions to the transport container were discussed in D4.2, alongside with the discussion of the layering structure to be used.

2.5 Synchronisation

The synchronisation in DIOMEDES is based on the fact that the video-view streams and the audio streams are encoded and multiplexed with the same System Time Clock (STC). Therefore, The PCRs, PTSs and DTSs of the streams originate from the same STC. The timing information (PCR) is contained in the DVB-T, and is used as a reference time to the User Terminal clock. The PCR provides a clock with a resolution of 27MHz and an accuracy of 500ns.

2.5.1 Synchronisation of DVB and P2P

AV Synchronisation & Demux & Buffer Module has to re-build the clock using the PCR from the DVB-T transport stream. This reconstructed new clock is streamed to the audio and video clusters. DVB-T streams are delayed in the buffer, to allow the parallel P2P video-views streams enough time to arrive. Even if only the DVB-T stream is used (since network is very congested), it should be delayed in the buffer to allow the clusters enough time for decoding and rendering before PTS time has arrived. The generated PCR clock should be delayed more than the buffer delay time. This method allows the video and audio clusters to get the streams before they should be displayed\heard according to the PTS. A detailed description of the synchronisation mechanism is given in Section 2.2.7. This mechanism can also be applied if there are other delivery methods in use like IP or DVB-S etc.

The P2P module receives periodic updates from the AV-Synchronisation module with its internal clock information (PCR). This is important during the whole session, but especially during the initialization, as the P2P client needs to find the adequate chuck to start from. Once the clock information has been received from the AV Synchronisation Module, the P2P updates the state of the session by modifying the requested clock parameter. At that instant, P2P first checks if it is falling behind the schedule or not. If the IP delivery is ahead of the DVB delivery, then no further steps are taken. But if the opposite situation occurs, the P2P video tries to estimate the first chunk it can download on time using two parameters: bitrate of the content and the current download rate.

2.5.2 Synchronisation of audio and video clusters

A dedicated PID only containing the PCR is used as synchronisation mechanism. The AV Synchronisation & Demux & Buffer Module sends this stream to the audio and the video clusters. PCR is interpreted as real time, at which the renderer plays back the frame with the corresponding PTS. Therefore the AV Synchronisation & Demux & Buffer Module has to send the media data early enough to allow decoding in time.

3 P2P NETWORK ARCHITECTURE

3.1 Overview

DIOMEDES P2P follows a similar architecture with the currently available solutions (namely Tribler and P2PNext) in terms of topology since DIOMEDES P2P is also mesh-based, in which peers pull data from a subgroup of peers in the network. However, there are also significant differences especially in the formation of data chunks and deployed adaptation strategies. In Section 3.2, some of Torrent-Based P2P video streaming solutions are reviewed and the problems with these approaches are underlined. In Section 3.3, the list of key differences from the mentioned projects is provided. Finally, in Section **Fehler! Verweisquelle konnte nicht gefunden werden.**, the key features of the developed P2P software are outlined. For the details of the P2P architecture, readers can refer to D4.5 and for the implementation details, readers can refer to D5.3.

3.2 Currently Available Solutions

State of the art P2P video streaming systems are mostly Torrent-based. Main examples of such systems are Tribler and Nextshare. Since the Torrent protocol employs fixed size chunks, these systems suffer from various limitations, which are described in the following. Moreover, these systems utilize a fixed sized chunk scheduling mechanism, which may cause heavier load over the seeder nodes.

Tribler is one of the most commonly used video sharing platforms that are compatible with the BitTorrent protocol. Tribler has some new features like a windowing mechanism to enable chunk scheduling in a timely manner, a social networking aspect to augment content retrieval and sharing (buddies), and novel concepts like give-to-get (a new incentive mechanism) [6]. According to its specifications, Tribler has no application layer concept because the chunks are filled with multimedia data in a chop-and-ship manner.

P2PNext is a European Union project that is based on the Tribler core [4]. Naturally, P2PNext is also BitTorrent compatible and follows its specifications. P2PNext is one of the first groups that target scalable video distribution over P2P networks and tries to augment the received video quality by performing rate adaptation. In NextShare (the name of video delivery platform), padding is used in order to align the group of pictures (GOP) boundaries to the fixed sized chunks. They have adopted Constant Bit-rate Coding (CBR) technique to create bitstreams at a certain rate such that padding does not cause much overhead [7].

Effects of Chop and Ship

In chop-and-ship packetisation, (which corresponds to mapping video bitstreams into Torrent chunks) NAL unit boundaries are disregarded and a NAL unit can be split into two distinct chunks. Since a half delivered NAL unit is useless, if a chunk is lost (or skipped due to the play-out deadline) it will not be possible to decode the other delivered half NAL unit (first NALU) in the following chunk. Moreover, even after skipping the half delivered NAL unit, it may still not be possible to decode the remaining bit-stream for a while due to prediction structure (e.g., if the I-frame is in the previous lost chunk). Tribler Protocol Specification states these problems and declares that buffer underruns can occur immediately after a re-buffering period [8]. The major cause of this problem is the P2P's inability to estimate the decodeable buffer duration and cannot determine the number of chunks it needs to receive before resuming the play-out during a re-buffing period.

Effects of constant bit-rate coding to reduce the amount of Padding

P2PNext performs adaptive streaming by discarding enhancement layers chunks when the buffer duration is low. Therefore, the ability to estimate the buffer duration correctly has a critical impact over the efficiency of adaptation. Consequently, P2PNext has a different approach in mapping the video stream to BitTorrent chunks.

P2PNext aligns the GOP boundaries with the chunks using padding. Padding is mandatory because due to the nature of video, it is impossible to have all GOPs in exactly the same length. In order to keep the padding ratio low, P2PNext utilizes CBR encoding. The side effect of this approach is described in the following.

CBR coding is mostly adopted in fixed rate channels and fixed sized mediums such as DVB and DVD, in which one cannot utilize the variations in the bitrate. In CBR coding, the quality of the video varies because the video output rate (the number of bits that is dedicated to a number of frames) is constant. In such cases, when the video has low motion or low complexity, the decoded video quality is higher and vice versa. Consequently, in order to keep the perceived video quality high, the video rate is commonly set to a high value in order to compensate the quality loss for scenes with high motion or high complexity.

A simple encoding test was performed to underline the adverse effects of using CBR coding in a channel with variable capacity like the Internet. In this test, the sample movie stream called `wildlife.wmv` that is provided freely in Windows 7 was used The video was encoded using the SVC extension of H.264/AVC using a single layer with no overhead of scalability. The content resolution is 1280x720 pixel. First, VBR coding with 5 different quantization parameters to represent 5 different video qualities (highest, high, medium, low, and lowest) were performed. Figure 8 depicts the change in quality (PSNR) over GOPs and the Figure 9 presents the corresponding bitrate of the sequences for the same interval.

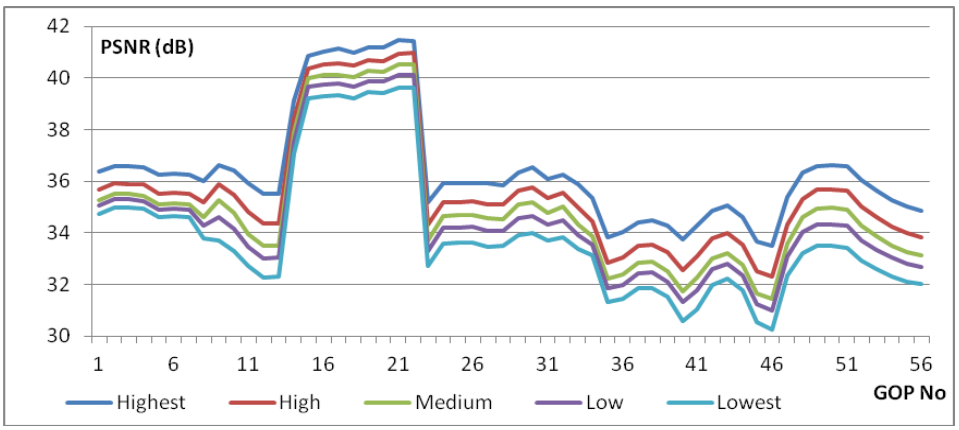

Figure 8 - PSNR over Time (1 GOP ~= 0.5 sec) Wildlife

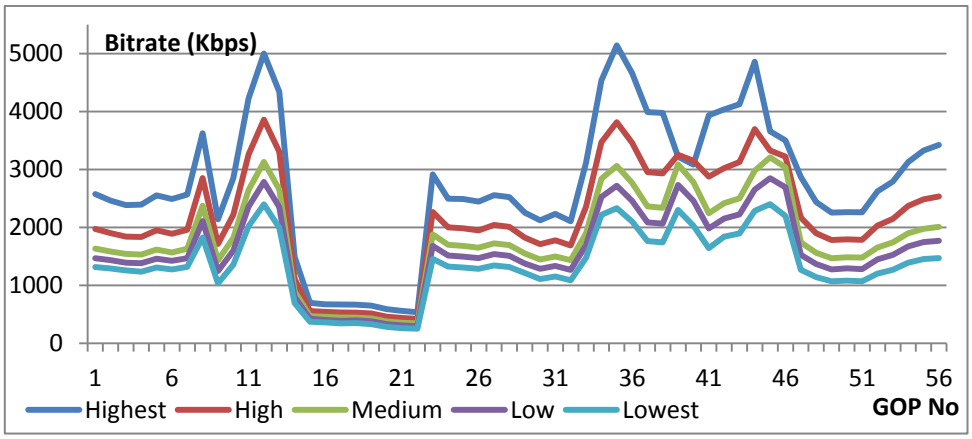

Figure 9 - Bitrate distribution over GOP

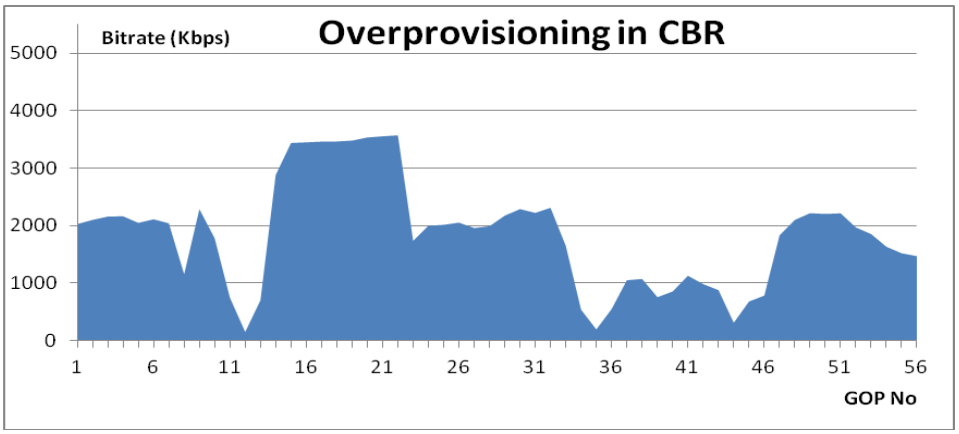

Figure 10 - The overhead of over provisioning in CBR

If this scene is to be encoded using CBR mode and if minimum acceptable PSNR value is 32dB then the high quality content (red line) can serve as a reference because in the high quality content, the minimum PSNR value is just over 32dB. Then the CBR coding rate must be about 4000Kbps which is the peak of the red line in Figure 8.

For such a case, the amount of the over–provisioned bit-rate to achieve the minimum PSNR requirement (i.e. 32 dB minimum) is depicted in Figure 10. Considering that the average rate of over provisioning is above 1.8Mbps, it is evident that CBR coding is not an optimum solution for video delivery over the Internet.

Considering the diminishing gain in the rate-distortion performance, even if the state of the art video codecs are utilized, the required bitrate to match certain quality levels are too high. For instance, if the minimum required quality is above 34 dB, then even 5000 Kbps is not enough. So, video streaming applications that use CBR coding have this hidden high bitrate overhead problem.

Using VBR Coding with Fixed Sized Chunks

Switching from CBR to Variable Bit-Rate (VBR) coding augments the padding overhead. Table 4 presents the overhead of padding when high quality contents are mapped to fixed sized chunks. These sequences are not static (monotonous), but comprise dynamic texture and motion, as well as scene changes. Considering that the overhead of padding comprises a

significant portion of the overall bit-stream, it can be concluded that switching to VBR coding is not a trivial solution of CBR coding problems for BitTorrent compliant P2P systems either. Moreover, when the chunk sizes are smaller than the GOP size, then multiple chunks are used to encapsulate a single GOP, which again creates again chunks that are not self *decodable*. The result from the previous section and this one reveals that regardless of the coding technique (CBR or VBR), using fixed sized chunks introduces inefficiency. In CBR, the inefficiency is due to the coding at high bitrates. In VBR, it is because of the padding.

	64K*	**128K***	**256K**
Wildlife	23.09%	48.03%	86.65%
Civ5 Trailer	24.01%	50.03%	87.43%

Table 4 - Padding overhead for different chunk sizes

(* Chunk sizes smaller than GOP sizes)

3.3 Differences of the DIOMEDES P2P from currently available solutions

Chunk Generation: Both Tribler and P2PNext use fixed sized chunks (to be compatible with Torrent) and discard application layer framing (ALF) concept, which is a very important aspect for video streaming over the Internet. Without ALF concept, video bit-streams are split into chunks regardless of GOP boundaries. Consequently, such P2P applications cannot estimate the duration of playable (decodeable) video in their buffers. In Tribler's protocol specifications this fact has been stated to cause frequent re-buffering during playback, when the chunks in the buffer are not useful due to prediction structure of the video. In the P2PNext case, the encoders are used in CBR mode, which is known to have a lower rate-distortion performance compared to VBR coding. Similarly, due to the usage of fixed sized chunks, padding is used to fit video packets into chunks. In DIOMEDES, chunks are formed using GOP boundaries. Therefore, it is possible to determine the duration of decodeable data in the buffer and perform adaptation more efficiently. It is believed that using application layer framing in a video streaming architecture is a significant contribution for video streaming over P2P applications. In the literature, there are numerous studies, which suggest that solutions using the ALF concept (cross-layer solutions) outperform the content agnostic transmission mechanisms. [10]

Adaptation: The adaptation of the content in case of multi-view is more complex than in the case of monocular (2D) video. In standard scalable coded 2D video, the order of streams' priorities is fixed (decreasing as it goes from base layer to the following enhancement layers). However, in the case of multi-view video, the stream priorities can change according to the user preferences, and the orientation of the scene. Therefore, DIOMEDES P2P follows more dynamic adaptation strategies mandated by the adaptation decision engine module (ADEM).

Security: The security mechanism in DIOMEDES covers both the data integrity and the user authentication using either symmetric or asymmetric keys. In the case of DIOMEDES P2P, the need of providing hash values and performing hash based data integrity checks are not required because it is possible to use the public key of the server to validate data originator

and symmetric key pair to check the data integrity. This means a significant reduction in the size of the metadata file.

Hybrid Model: In DIOMEDES, the content is pre-processed and pre-encoded to be played-out later in synchronicity with the DVB-T channel. Obviously, it is possible that the content available in the Main Seed Server can be distributed (with a dedicated time offset) before the DVB play-out. In such cases, it is possible to distribute the video content in an intelligent way to minimize the server load (see D4.5). This also creates an incentive for the content provider to prepare the content ahead of time to deliver large amount of data at a low cost. Naturally, the P2P policies that are adopted in DIOMEDES do not rely on the differences between the content preparation and play-out deadline. If the deadline is close for chunk download, necessary adaptations are performed to deliver more recently uploaded content. If there is time, then the system uses an intelligent way of distributing the content seamlessly by consuming significantly less bandwidth and making the distribution cheaper and safer. Therefore, the chunk scheduling policies in DIOMEDES always keeps an eye on the bandwidth requirements of the main seed server.

3.4 Adaptive streaming in the DIOMEDES P2P

A new streaming mechanism is proposed with distinguished features from the abovementioned platforms that performs adaptive P2P video streaming. In the proposed design, contents are encoded using scalable video coding (SVC) extension of H.264/AVC and base and enhancement layer chunks are created, which can have arbitrary length. On top of that, a variable size windowing mechanism is adopted to minimize the bandwidth requirement of the seeders and maintain a scalable solution against increasing number of leechers. [5].

Variable Size Chunks using SVC H.264/AVC

The limitations of fixed size chunks were discussed in section 3.2. To this effect, variable length chunks are allowed, one for each GOP. There are two types of chunks:

- base layer chunks are obtained by merging base layer NAL units for a GOP which are prioritized and
- enhancement layer chunks that are obtained by merging all enhancement layer NAL units for a GOP. These are discardable in case of bandwidth scarcity.

Dynamic Scheduling Window

A scheduling window is used in many propositions to enable multimedia streaming for BitTorrent based overlays (including Tribler and NextShare) [1]-[4]. The scheduling window restricts chunk picking policy to choose from a limited number of chunks that are close to the play-out deadline. However, this restriction also decreases the P2P activity (chunk exchange among leecher, assuming that seeders are content providers) because the peers are less likely to have distinct chunks. In the standard BitTorrent case, a peer chooses chunks that are least distributed in the network (rarest first). In short, a small window size augments the likelihood of uninterrupted video delivery, whereas a large window size increases the rate of chunk exchange among peers and decreases the load on the seeders.

A dynamic window is proposed that has varying length, which tries to grasp both abovementioned aspects as much as possible. For this purpose, the size of the window is determined each time it slides, based on the current duration of the video buffer. The new window size is calculated using (1) which correlates the size of the new window in number of chunks (w_s) to the current buffer duration (b_d). The buffer duration is the time interval that can

be spent to download the next window. The number of bytes that should be downloaded is equal to the number of chunks in the next window (w_s) multiplied by the average size of chunks (provided in the metadata file) and divided by the average download rate. So, w_s is the only unknown at a point, where it is wanted to calculate the next window size. In the case of floating point, the window size in chunks is rounded to the nearest integer.

$$b_d = \frac{w_s \times c_s}{d_r} \qquad (1)$$

w_s	window size in # of chunks
c_s	average chunk size
d_r	download rate
b_d	buffer duration

The thorough evaluation of these features can be found in more detail in D4.5 and D5.3.

4 RISK ASSESSMENT

The following list describes possible risk scenarios, their respective mitigation strategies and risk levels. The risk scenarios are consecutively numbered and the numbers after the dash mark the affected work packages.

- *R 1 – WP 5/6: Real-time decoding of media cannot be achieved for the demonstrator*

  Mitigation strategy:

  If it cannot be achieved, then the "medium term" scenario will be demonstrated, rather than the "long term" scenario (See DoW beginning of section B 1.1). To increase the chance that it can be achieved, the total number of viewpoints simultaneously decoded will be decreased (e.g. to two), at the expense of lower viewpoint resolution. Note that this risk applies mainly to the video. No problems are anticipated with audio.

  Risk level: Medium

- *R 2 – WP 3/5/6: 3D video quality is unacceptable on the 3D displays to be used in the project*

  Mitigation strategy:

  The manufacturer of the display that is found to provide insufficient 3D video quality will be consulted to determine whether improvements in rendering can be made.

  Risk level: Medium

- *R 3 – WP 4/5/6: IP network insufficient for supporting 3DTV streaming to end users*

  Mitigation strategy:

  The 3D media streaming server will be made adaptable to varying bandwidth constraints. In the worst case, the server will be located at the demo site on a non-public network.

  Risk level: Low

- *R 4 – WP 5/6: DVB-T test facilities no longer available at IRTs site*

  Mitigation strategy:

  UNIS DVB-T emulators will be used in place of the real facilities. Alternatively, UNIS Rhode & Schwarz DVB-S lab will be used to emulate the transmission over DVB-S.

  Risk level: Low

- *R 5 – WP 3/4: Audio-visual attention models do not work consistently on all of the 3D media acquired for D3.1*

  Mitigation strategy:

  In this case, the attention models, developed during Task 3.3 will be optimized for particular types of content (e.g. a particular sport type) for the purpose of the demonstrator (D6.4). The content metadata will be used to select between different attention models, developed for different types of content (Note: Content

acquisition is described in D3.1, while the attention models will be reported in D3.4)

Risk level: High

- *R 6 – WP 3/4: Quality of Experience models have weak correlation with results from listening/viewer tests.*

  Mitigation strategy:

  Existing 2D and stereoscopic quality assessment techniques, such as those recommended by the ITU (e.g. ITU-R BT-500 and ITU-R BT.1438), that are known to work well for 2D videos can be adapted to estimate the experience score

  Risk level: Low

- *R 7 – WP 3/6: Holografika's display is not of sufficient quality for end-user*

  Mitigation strategy:

  Experiments will be carried out with a variety of displays that are already known to meet end-user requirements.

  Risk level: Medium

5 CONCLUSIONS

This document presented the final architecture of DIOMEDES as an end to-end-system, based on the initial and interim reference system architectures described in deliverable D2.1 and D2.2, and the work done in the other related work packages. All modules (on the server and the user terminal side) were described and modified with respect to the progress in the other work packages. More detailed information for all system components can be found in the related WP3, WP4 and WP5 deliverables.

REFERENCES

[1] ISO/IEC 13818-1, MPEG-2 Systems
[2] A. Vlavianos, M. Iliofotou, and M. Faloutsos, “BiToS: Enhancing bittorrent for supporting streaming applications,” in Proc. IEEE Global Internet, Apr. 2006
[3] J. Pouwelse, P. Garbacki, J. Wang, A. Bakker, J. Yang, A. Iosup, D. H. J. Epema, M. Reinders, M. van Steen, and H. Sips, “Tribler:A social-based peer-to-peer system. In Proc. of IPTPS,” 2006
[4] Capovilla, et. al., “An Architecture for Distributing Scalable Content over Peer-to-Peer Networks”, in Proc. of Int. Conf. on Advances in Mult. (MMEDIA’10), Athens, Greece, 2010
[5] K. Aydogdu, E. Dogan, H. Gokmen, S. Savas, C. Gurler, J. Hasselbach, T. Korn, T. Adari, “DIOMEDES: Content Aware and Adaptive Delivery of 3D Media over P2P/IP and DVB-T2,” NEM Summit 2011, Torino Italy, 27-29 September 2011
[6] J. Pouwelse, P. Garbacki, J. Wang, A. Bakker, J. Yang, A. Iosup, D. H. J. Epema, M. Reinders, M. van Steen, and H. Sips, “Tribler:A social-based peer-to-peer system. In Proc. of IPTPS,” 2006
[7] M. Eberhard et al., “Knapsack problem-based piecepicking algorithms for layered content in peer-to-peer networks,” in ACM Workshop on Advanced Video Streaming Techniques for P2P Networks and Social Networking, 2010, pp. 71–76
[8] A. Bakker et al. “Tribler Protocol Specification” http://svn.tribler.org/bt2-design/proto-spec-unified/trunk/proto-spec-current.pdf, 1 2009.
[9] Philips 3D Solution, “User Manual – 42” 3D Display”, http://www.business-sites.philips.com/3dsolutions/servicesupport/docs/docs.page, last accessed 18 June 2010
[10] S. Wenger, “H.264/AVC over IP,” IEEE Trans. Circuits Syst. Video Technol., vol. 13, pp. 645–656, July 2003

APPENDIX A: GLOSSARY OF ABBREVIATIONS

A	
AAC	Advanced Audio Coding
ADEM	Adaptation Decision Engine Module
AES	Advanced Encryption Standard
ALF	Application Layer Framing
A/V	Audio / Video
AVC	Advanced Video Coding
D	
DHT	Distributed Hash Table
DPX	Digital Picture Exchange
DTS	Decoding Timestamp
DVB	Digital Video Broadcasting
DVI	Digital Video Interface
E	
ES	Elementary Stream
ETSI	European Telecommunications Standards Institute
F	
FIFO	First In First Out
G	
GUI	Graphical User Interface
GOP	Group of Pictures
H	
HDMI	High Definition Multimedia Interface
HDTV	High Definition Television
HTTP	Hypertext Transfer Protocol
I	
I/O	In / Out
IP	Internet Protocol
ITU	International Telecommunication Union
J	
JSON-RPC	JavaScript Object Notation – Remote Procedure Call
JVT	Joint Video Team
K	
KPI	Key Performance Indicator
L	

LGPL	Lesser General Public Licence
LTC	Longitudinal Timecode
M	
MD-SMVD	Scalable Multiple Description 3D Multi-View Video
MOS	Mean Opinion Score
MPEG	Motion Picture Experts Group
MPEG3DAV	Motion Picture Experts Group 3D Audio-Visual
MVC	Multi-view Video Coding
MVD	Multi-view Video plus Depth
N	
NAL	Network Abstraction Layer
NPP	Native Pixel Parallax
NSA	National Security Agency (USA)
O	
OFB	Output Feedback Mode
P	
P2P	Peer to Peer
PCR	Program Clock Reference
PES	Packetized Elementary Stream
PID	Program Identifier
PLR	Packet Loss Rate
PTS	Presentation Timestamp
Q	
QoE	Quality of Experience
S	
SDTV	Standard Definition Television
SHA	Secure Hash Algorithm
SMPTE	Society of Motion Picture and Television Engineers
SNR	Signal to Noise Ratio
SOAP	Service Oriented Architecture Protocol
STC	System Time Clock
SVC	Scalable Video Coding
T	
TCP/IP	Transmission Control Protocol / Internet Protocol
TS	Transport Stream
TV	Television
U	

UDP	User Datagram Protocol
V	
VLC	Video Lan Client
W	
WFS	Wave Field Synthesis