# Initial Solution for Scalable and Adaptive Coding

## Project Details

<table>
<thead>
<tr>
<th>Project Number:</th>
<th>Project Acronym:</th>
<th>Project Title:</th>
</tr>
</thead>
<tbody>
<tr>
<td>045461</td>
<td>MING-T</td>
<td>Multistandard Integrated Network convergence for Global Mobile and Broadcast Technologies</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Contractual Delivery Date:</th>
<th>Actual Delivery Date:</th>
<th>Deliverable Type* - Security**:</th>
</tr>
</thead>
<tbody>
<tr>
<td>30/09/2007</td>
<td>20/11/2007</td>
<td>REPORT, PUBLIC</td>
</tr>
</tbody>
</table>

* Type: P – Prototype, R – Report, D – Demonstrator, O – Other  
** Security Class: PU- Public, PP – Restricted to other programme participants (including the Commission), RE – Restricted to a group defined by the consortium (including the Commission), CO – Confidential, only for members of the consortium (including the Commission)

## Responsible and Editor/Author:

<table>
<thead>
<tr>
<th>Nicola Conci</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Organization:</th>
<th>Contributing WP:</th>
</tr>
</thead>
<tbody>
<tr>
<td>CREATE-NET (CN)</td>
<td>WP3</td>
</tr>
</tbody>
</table>

## Authors (organizations):

CREATE-NET

## Abstract:

The aim of this document is to propose the initial solution to exploit scalable video coding and hierarchical modulation in the context of the MING-T project. In particular, beside an overview of the main features of the selected video coding standard (SVC – scalable implementation of H.264), the document contains some valuable features that represent an added value to the provisioning of video service, enhancing the efficiency and flexibility of the whole system.

## Keywords:

Scalable coding, SVC, RTP, H.264, motion analysis, video streaming
Revision History

The following table describes the main changes done in the document since created.

<table>
<thead>
<tr>
<th>Revision</th>
<th>Date</th>
<th>Description</th>
<th>Author (Organization)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>29/08</td>
<td>Initial Draft</td>
<td>Nicola Conci (CN)</td>
</tr>
<tr>
<td>0.2</td>
<td>03/09</td>
<td>Second Draft for circulation in CN</td>
<td>Nicola Conci (CN)</td>
</tr>
<tr>
<td>0.3</td>
<td>04/09</td>
<td>Draft revision</td>
<td>G.M. Maggio (CN)</td>
</tr>
<tr>
<td>0.4</td>
<td>25/09</td>
<td>Comments on H.264 SVC decoder</td>
<td>Jingyuan Wang (THU)</td>
</tr>
<tr>
<td>0.5</td>
<td>26/09</td>
<td>Final Draft for circulation among partners</td>
<td>Nicola Conci, Antonio Matera and Yabin Ye(CN)</td>
</tr>
<tr>
<td>0.6</td>
<td>26/09</td>
<td>Spellcheck</td>
<td>Nicola Conci</td>
</tr>
<tr>
<td>0.7</td>
<td>12/10</td>
<td>Integration with spatial scalability and FGS tests</td>
<td>Nicola Conci and Antonio Matera</td>
</tr>
<tr>
<td>0.8</td>
<td>16/10</td>
<td>Final Draft</td>
<td>Nicola Conci, Antonio Matera, Norman Hendrich, Lifeng Sun, Wang Jingyuan</td>
</tr>
<tr>
<td>1.0</td>
<td>12/11</td>
<td>Document Finalization</td>
<td>Nicola Conci, Antonio Matera, Norman Hendrich, Wang Jingyuan</td>
</tr>
</tbody>
</table>
# Table of Contents

1. **Abstract** .............................................................................................................................................. 4  
2. **Introduction and Overview** .................................................................................................................. 5  
3. **Scalable Coding in H.264** ...................................................................................................................... 6  
   3.1 Scalable Coding Architectures ............................................................................................................. 6  
   3.2 General description and short review of AVC and SVC ................................................................. 9  
   3.3 Scalable coding and unequal error protection .................................................................................. 11  
4. **MING-T Scalable-Video Architecture and Components** ..................................................................... 13  
   4.1 Components definition ..................................................................................................................... 13  
   4.2 SVC in the DVB-H framework ........................................................................................................ 17  
   4.3 SVC in DTMB ...................................................................................................................................... 17  
   4.4 Pre-stored vs. live video transmission .............................................................................................. 18  
   4.5 Motion analysis and scalability adaptation ...................................................................................... 18  
5. **Experimental results** ................................................................................................................................ 19  
   5.1 General considerations and work organization ................................................................................ 19  
   5.2 Encoder rate control .......................................................................................................................... 20  
   5.3 Encoder performance on example sequences .................................................................................. 21  
   5.4 Encoding for un-equal error protection .......................................................................................... 23  
   5.5 Stream-format and RTP-packetizing ................................................................................................. 26  
   5.5.1 Single NAL Unit Packet ............................................................................................................... 29  
   5.5.2 Aggregation Packets .................................................................................................................... 29  
   5.5.3 Fragmentation Units (FUs) .......................................................................................................... 30  
   5.6 Decoder .............................................................................................................................................. 32  
   5.6.1 VC Stream Extraction .................................................................................................................... 32  
   5.6.2 Cross-Layer Synchronization by using RTP ............................................................................... 34  
   5.6.3 Performance Test of JSVM Decoder ............................................................................................ 34  
6. **Next steps** ............................................................................................................................................... 37
1. ABSTRACT

The scope of this document is to present initial solutions for the integration of scalable video coding formats within the MING-T project framework. In the following pages the recent advances in scalable video coding are presented and the potential applications in MING-T are discussed. The document covers the most important issues related to the implementation of a scalable system in the project architecture, trying to identify the gaps between the state of the art and the actual requirements, and explaining the implementation activities carried out so far.

For background information about the project including an overview and publications please visit the project website at http://www.ming-t.org/.
2. INTRODUCTION AND OVERVIEW

Over the last few years, the transmission of audio-visual information across wireless channels has become one of the most challenging applications. This interest in video communication has grown thanks to the availability of more powerful and effective transmission techniques and protocols and, at the same time, source coding techniques that allow the delivery of a video (uncompressed requiring huge amounts of data) with considerably reduced payload burden. In the framework of the DVB-H standard, the video signal is sent over a wireless link whose channel capacity is not sufficient to carry the same amount of data as required by the MPEG-2 codec adopted by the terrestrial DVB-T. Thus, the transmission of the video calls for more effective compression techniques. One of the most advanced coding standards has been identified in the H.264 codec, jointly developed by the ISO-IEC and ITU-T standardization bodies. H.264 has been designed to allow the implementation of a wide range of applications starting from low bit-rate videos (40-50 kbps) to high-quality videos (>1 Mbps). In practice, H.264 exhibits its best performance on low/medium rate streams, because it is able of reducing the amount of data to be delivered while only slightly reducing the quality. The gap between H.264 and the previous coding standards consists in the insertion of a series of new features that increment the compression efficiency but also the computational complexity. More detailed information about the standard will be given in chapter 3 of this document.

As MING-T aims at enabling the interoperability between the DVB-H [4] standard and the Chinese standard DTMB [2][3], the integration is quite a challenging task even at the application level (video communication), as it involves many different layers of the ISO-OSI stack. In particular, not only the video coding layer (VCL) must be taken into account, but also the network aspects play a major role in this context.

This document is organized as follows: Section 3.1 gives a general introduction to the problem of scalability and explains why we are adopting a scalable coder instead of a transcoder. In sections 3.2.1 and 3.2.2 the most important features of the non-scalable and scalable versions of the chosen codec H.264 are presented, while section 3.3 describes some common principles of scalable-coding in video applications for wireless transmission and unequal error protection.

Chapter 4 is focused on the MING-T framework and will present the overall architecture of the project from the video perspective, highlighting the core components which will be developed in the project.

The experimental results from the ongoing research are presented in chapter 5. First, section 5.1 lists some general considerations regarding the experiments and implementation, while section 5.2 concentrates on the encoder bit-rate control. Next, section 5.3 focuses on the performance of the reference-encoder, while section 5.4 describes the results obtained with un-equal error-protection. Section 5.5 explains the RTP-packetizing format implemented so far. Finally, section 5.6 describes the decoder, including stream-extraction and cross-layer synchronization.

Chapter 6 summarizes the report and lists the next steps planned towards achieving the final project prototype and demonstrations.
3. SCALABLE CODING IN H.264

3.1 Scalable Coding Architectures

Scalable video coding has been introduced almost at the same time for both MPEG-2 (standardized by the ISO/IEC body) and H.263 (ITU-T). The advantage of having a scalable architecture allows for a more flexible and efficient way of transmitting visual content across different channels. In fact, the fundamental principle of a scalable coder is to “encode once and decode many”, overcoming the problem of transcoding [4] when dealing with heterogeneous terminals and networks. Before the introduction of the scalability principle, there was the need of preparing several different video sources in order to fulfill the requirements of different categories of users. In particular, when transmitting a video over the Internet, the bitrate of the sequence could be too high for certain users (maybe using a common 56kbps modem) and on the contrary, if a low rate is used to fit the needs of all users, many of them could experience a low quality although paying a higher fee for the requested service. In order to cope with the above mentioned issues, traditionally transcoding has been considered the most efficient way to provide different quality of service to different users. However, the natural implication of transcoding is that the obtained bit-rate is again suitable for a single user only. At the same time, the process requires a stack of transcoders to provide all users simultaneously with the same visual content. This process requires an ad-hoc conversion (see transcoders T1, T2 and T3 in Figure 1) each time a stream needs to be sent over the network. Typical values for the final bitrates of the single streams T1, T2, and T3 could be 150-200kbps, 384-640kbps, 1.5-2Mbps respectively.

![Diagram of transcoding-based video server](image)

**Figure 1. Example of transcoding-based video server. Ad-hoc streams are generated for each user category.**

As an alternative, scalable video coders provide coding strategies and formats able to limit the amount of data to be processed in order to support fast conversion and rate adaptation via simple dataflow-selection procedures. In this context, techniques such as layered video, fine granularity or bit-plane coding, and multiple description coding could represent helpful alternatives to classical transcoding approaches, providing high flexibility and reusability of the encoded stream. This makes it possible to optimize the adaptation layer as well, creating complementary streams able to easily represent the visual content. The structure of a scalable coder acts on a different level if compared to a transcoder, for it is a source coder that directly produces an adaptable network-ready stream, not requiring additional layers and operations. In this case, of course, part of the complexity is left to the decoder, which should be able to interpret and
manage a structured video stream (layered, progressive, multi-source, etc.) and extract the best possible subset of data for decoding (figure 2).

In order to facilitate the comprehension of the scalable architectures, let’s see in the following how we can implement a scalable coder. Namely there are three ways to achieve scalability: temporal (figure 3), spatial (figure 4) and quality or signal-to-noise-ratio (SNR, figure 5). The so-called fine-grained scalability (FGS) is a particular implementation of SNR coding, where the enhancement layer is coded bitwise in order to allow smooth variations in the bitrate, instead of an on-off layer concept (figure 6) [6].

Figure 2. Example of a scalable coder-based video server. Each user decodes the best quality stream according to the capabilities of his client.

Figure 3. Temporal scalability. In the example, the base-layer uses I-frames (red) and P-frames (green) at one-third of the framerate of the high-quality (base+enhancement) stream.
Figure 4. Spatial Scalability uses layers of different image resolution.

Figure 5. Quality Scalability.

Figure 6. Traditional quality (SNR) vs. fine-grained scalability (FGS).
3.2 General description and short review of AVC and SVC

In this section the overall architecture of the MING-T video transmission is described. For better understanding of the problems and the different scenarios, a brief review of the AVC [7] [8] and SVC [9] standards will be given in the following subsections.

3.2.1 Main features of the AVC coding standard

The introduction of the H.264 coding standard (commonly referred to as 14496 part 10 in the ISO-IEC nomenclature) sets a new paradigm of video coding, especially designed for very low bit-rate coding. The codec is also known as AVC (Advanced Video Coding). Simulation results demonstrate that the compression gain with AVC can be up to 50% higher than achievable with previous compression algorithms at the same visual quality. This is possible thanks to different features that have been introduced in the standard definition. The most important innovations of AVC can be summarized as follows:

- An adaptive deblocking filter to reduce the blocking artifacts has been introduced in the prediction loop. The output of the filter can then be used to predict the forthcoming macroblocks.
- Instead of using only a single frame for the prediction, AVC can use multiple frames to be stored in the memory for a more accurate prediction.
- In H.264/AVC it is also possible to implement a scheme for Intra prediction. In practice the transmitted intra-blocks can be used to predict other blocks belonging to the same frame.
- The Discrete Cosine Transform (DCT) used in former standards is replaced by an integer transform, making the code execution much faster and with a reduced impact on the final quality.
- The whole coding algorithm was conceptually split into two different parts, the VCL (Video Coding Layer) and the NAL (Network Abstraction Layer). The VCL is related to the actual algorithms for video compression, while the NAL is targeted at preparing the stream for the delivery over a packet-oriented network. The use of the NAL allows a seamless integration with both previous standards like MPEG-2 and TCP/IP networks. Further information about this aspect can be found in [8].

3.2.2 Main features of the SVC coding standard

SVC represents the scalable extension of AVC. The standardization process of the codec has not been finalized yet, although a wide range of functionalities can be already considered as a stable and consistent part of the emerging standard. A schematic diagram of the codec is shown in Figure 7.

As expected from a scalable coder, SVC includes temporal, spatial and quality (SNR) scalability. One of the important advantages of SVC is that the lowest layer (i.e., the base layer with lowest quality) is compliant to the non-scalable coder (namely H.264/AVC), thus ensuring some level of backward compatibility. Even though the graphical representation shown in Figure 7 only includes spatial and quality scalability (temporal downsampling is not reported for simplicity), the diagram demonstrates the efficiency and the flexibility of the current implementation. It allows for multi-resolution and variable-quality coding, letting the user choose the most appropriate version according to his terminal capabilities. The whole scalability process has been revised in H.264/SVC and some innovative aspects have been added in order to enhance efficiency and performance. Figure 7 represents a very general implementation of the
codec, demonstrating its capability in dealing with a large number of layers defined by a combination of different scalability techniques.

Figure 7. General design of the SVC encoder.

As far as temporal scalability is concerned, SVC exploits the innovations introduced in H.264/AVC to achieve a layered stream by using a hierarchical prediction scheme. This principle establishes that every frame can be considered as a reference picture, usually following a dyadic hierarchical tree. In this way it is possible to start from a coarse temporal resolution, while quality can be progressively refined by adding the previously frames discarded following the tree. An example with a 4 dyadic tree is reported in Figure 8.

Figure 8. Temporal Scalability in SVC with hierarchical dyadic prediction.
Further details about the SVC features can be found in [9]. As in the non-scalable version of the H.264 codec, the standard implements two different file formats, the Annex B and the RTP format. The first one is mainly suitable for data storage, while the second one is targeted at delivering the stream over a UDP/IP network. As usual in streaming applications, the latter format is based on the RTP headers and timestamps to efficiently assembly the transmitted video at the decoder. The application of a pre-defined header at the network level is not only important for delivering having a fixed timestamp (in fact the stream could be even split regardless of the contained data), but the target is to introduce a set of breakpoints in the file in order to allow the best reconstruction even when a certain number of packets is lost during transmission. In SVC the rules for packet fragmentation and aggregation are still under definition, and the expected final document will somehow reflect the ones defined for AVC in [12] which in turn are based on [11]. Regarding explicitly the implementation of the RTP payload format in SVC, so far only an IETF draft is available in [13].

Although SVC represents probably the most flexible scalable codec that has been presented so far, there are some practical limitations that make it difficult to exploit the whole set of features. In particular the dyadic prediction (although very effective from a design point of view) can be sometimes too heavy, requiring long times for encoding and decoding. At the same time, a large number of FGS layers turns out to be too complex in some situations, and a compromise needs to be met. In the next section, some simple examples on how scalability techniques can be adopted and used in wireless transmission are illustrated.

### 3.3 Scalable coding and unequal error protection

The framework of scalable coding is well suited for delivering video to heterogeneous terminals and networks. An important limitation of this scenario is the need of correctly decoding the base layer. In fact, if the base layer is not fully available at the decoder, the enhancement layer becomes useless and the user cannot enjoy the video. Therefore at least the content in the base layer must experiment a higher level of protection when transmitted, in order to ensure its correct decoding. The protection can be achieved by exploiting both source and channel coding, or if necessary even a joint source/channel coder. This concept falls into the definition of the so-called Unequal Error Protection (UEP), whose target is to protect the bitstream in a differentiated manner, according to the priority of each layer. Even though several different approaches can be employed to develop a UEP system, many of them rely on efficient bandwidth allocation techniques (e.g. CDMA, MC-CDMA, OFDMA) as described in [14][15].

Another very interesting method to deliver multimedia contents over wireless channels is presented in [10], where the so-called Hierarchical Modulation (HM) is used to provide a higher protection to the base layer. The objective of hierarchical modulation is to embed the most important part of the visual content (namely the base layer) into the most significant bits of the modulation technique while the pieces of information that can eventually be discarded are assigned to the less significant bits. The approach has been introduced as an option in the DVB-H standard, and the basic concept is having at disposal a QPSK in a 64QAM (or 16QAM). In the 64 QAM example, 6 bits are required for each symbol; the most significant 2 bits (called the High Priority stream) are used to transmit the base layer (therefore ensuring a higher protection), while the remaining 4 bits (assigned to the Low Priority stream) are left for the enhancement layer (at a double rate).

A graphical representation of the basic principle of hierarchical modulation is presented in Figure 9, directly retrieved from the official DVB-H documentation.
Figure 9. Hierarchical Modulation in DVB-H
4. MING-T SCALABLE-VIDEO ARCHITECTURE AND COMPONENTS

4.1 Components definition

This section is devoted to the video coding aspects, in order to satisfy the MING-T requirements. The overall architecture of the project, from the video coding perspective is shown in Figure 10, where the terms HP and LP refer to the High Priority stream and Low Priority stream of the hierarchical modulation, respectively.

As illustrated in Figure 10, the video to be transmitted can be either generated from a live source (like for example a common webcam) or from a pre-stored video sequence (like movies or offline transmission). This aspect will be described in more detail in section 4.4, as the features of the two encoders must respect different constraints in terms of functionalities and encoding time. After the video is coded (or retrieved from a file server), it is then sent over the wireless channel (DVB-H or DTMB) according to the available transmission features.

In particular, when the DVB-H framework is adopted, the video can be transmitted by employing the hierarchical modulation. This allows transmitting a scalable sequence with the base layer delivered on the high priority channel, while the enhancement layer can be transmitted on the low-priority channel at a double bitrate. In this case, the SVC stream must be divided into two different sub-streams. The first sub-stream only includes base layer, the other sub-stream includes the enhancement layer. The base layer is transmitted over the HP stream of DVB-H, and the enhancement layer is transmitted over the LP stream. So far, hierarchical modulation is not available in DTMB, and therefore this scalability-option cannot be used to transmit video data over DTMB. If we use the DTMB standard, both SVC streams should be merged into one single stream, and transmitted over the broadcasting channel. This scenario is illustrated in Figure 10.
To decode the SVC streams transmitted over the DVB-H channel, we must first combine the base and enhancement layers into one stream. As shown in Figure 11, after the DVB-H demodulator demodulates the DVB-H broadcasting signal into sub-streams, a sub-streams combiner is required to reconstruct the complete SVC stream. If the broadcasting signal is the DTMB signal, the DTMB demodulator will demodulate the signal into a full quality SVC stream.

Figure 11. Overall architecture of the video decoder part.

To solve the heterogeneity problem of terminals in a video application system, different kinds of videos must be supplied different video services. In a traditional video-on-demand (VoD) application scenario based on scalable coding, the video server gives each user an individual video stream to support the video applications in terminal. During a service period, the terminal first sends a request message to the video server. This message includes the requested content, bit-rate, frame-rate, screen-resolution and other information of the required stream. According to this information, the server then extracts a suitable video stream from the SVC stream, e.g. by cutting or selecting different enhancement layers to complement the base layer. Because each terminal receives an individual service stream in this VoD application scenario, the task of stream extraction can be done by the SVC server. Using different extraction parameters to generate the different streams solves the heterogeneity problem of terminals. This structure of video application scenario is shown in figure 12.

On the other hand, the basic video delivery scenario considered within project MING-T is instead a broadcasting system. In this scenario, all terminals receive the same video stream. Therefore, the VoD application pattern described above can’t be used. If we would leave the stream extraction work for the server, the system can only meet the network’s requirements; the requirement of the different terminals cannot be met, see Figure 13.
Figure 12. VoD Scenario.

Figure 13. Broadcasting Scenario.
Table 1 lists the main components related to the video encoding and transmission, together with the project partner responsible for its implementation.

<table>
<thead>
<tr>
<th>Component</th>
<th>Responsible partner</th>
<th>Component Description</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Camera</td>
<td>CN</td>
<td>This module is an API interface that connects the camera (acquisition module) to the encoder</td>
<td>The implementation requires the integration in the encoder of a capturing system, which will be implemented using common Open Source libraries like OpenCV</td>
</tr>
<tr>
<td>Offline Encoder</td>
<td>CN</td>
<td>This encoder includes all the needed features. It does not need to work in real-time as it is used for demonstration purposes. Therefore videos are processed offline and transmitted separately</td>
<td>The encoder part includes the definition of the RTP interface to feed into ENS’s IPE</td>
</tr>
<tr>
<td>Live Encoder</td>
<td>CN</td>
<td>The live encoder is aimed at transmitting a low resolution and low framerate video, with reduced scalability functionalities, as the computational complexity of real-time encoding cannot be met with the current version of the JSVM software</td>
<td>During the discussion among partners, it was agreed that a stable system with robust real-time decoding is more important than the real-time encoder.</td>
</tr>
<tr>
<td>Decoder</td>
<td>THU</td>
<td>The decoder (installed on the terminal) is responsible of decoding the received video sequence in real-time for presentation to the user. In case DVB-H is used, the decoder merges the High and Low Priority streams into a single file before decoding</td>
<td>The decoder part includes the definition of the SVC RTP file format to extract the video data.</td>
</tr>
<tr>
<td>DVB-H Modulator</td>
<td>ENS</td>
<td>The modulator receives the scalable video as input and delivers it over the wireless link exploiting scalable coding and hierarchical modulation</td>
<td></td>
</tr>
<tr>
<td>DTMB Modulator</td>
<td>THU</td>
<td>The DTMB modulator is only able to deliver the base layer of the encoded file as hierarchical modulation is not supported yet</td>
<td></td>
</tr>
<tr>
<td>DVB-H Demodulator</td>
<td>FS</td>
<td>The demodulator receives the modulated stream and gives the raw video data to the decoder</td>
<td></td>
</tr>
<tr>
<td>DTMB Demodulator</td>
<td>THU</td>
<td>This demodulator receives one video layer and passes the data to the SVC decoder (working in AVC mode)</td>
<td></td>
</tr>
<tr>
<td>Terminal Player</td>
<td>THU- NSN - BUPT</td>
<td>The player should be able to interface with the decoder in order to display the video sequence.</td>
<td></td>
</tr>
<tr>
<td>Motion Analyzer / Scalability Switcher</td>
<td>CN</td>
<td>The motion analyzer selects the best scalability technique to be adopted for the transmission, according to the motion which is present in the sequence. This mode is only possible for offline encoding due to the high computational complexity.</td>
<td></td>
</tr>
</tbody>
</table>
4.2 SVC in the DVB-H framework

From the communication perspective, one of the most challenging features in the DVB-H standard is the capability of the so-called hierarchical modulation. As already explained in section 3.3 above, this option is a perfect match for the concept of video scalability available in SVC. The use of the hierarchical modulation allows distributing the amount of data over a double range of bitrate, where the base layer is delivered on the HP (High priority) link, while the enhancement layer is left to the LP (Low priority) link.

It remains to decide which of the scalability options of SVC (temporal, quality, spatial) are to be chosen for the transmission, given the characteristics of a DVB-H channel. Of course, the above scalability types can be used independently or in combined configurations.

The application scenario of project MING-T refers to video broadcast and wireless communication with handheld mobile user terminals such as PDAs or similar devices. Therefore, we can assume a small screen size and a typical screen resolution like QVGA (320x240 pixels) or CIF (352x288 pixels). In other words, the typical screen resolution is usually fixed. This allows to temporarily discard the spatial scalability feature and concentrate on the temporal and SNR scalability instead. Therefore the objective of the ongoing research is to jointly exploit FGS and temporal scalability depending on the selected encoding mode (live/offline). Further explanations about this aspect are provided in section 4.4 and 4.5.

4.3 SVC in DTMB

At the moment, DTMB does not provide hierarchical modulation as an option, and therefore the application of a scalable codec is not straightforward. For example, project MING-T also considers handovers between different broadcasting networks. In the case of a network-handover from DVB-H to DTHB, no one-to-one mapping of the HP and LP layers is possible, and a suitable stream adaptation is required. Anyway, this does not mean that no integration can be performed. In fact, the simplest way to adapt the stream would be to deliver only the HP stream over the DTMB channel. The overall quality of the received layer can be considered as satisfactory in any case, because the PSNR remains on average above 30dB (for details see chapter 5). This scheme was already shown in Figure 10 and Figure 11.

Together with the above-mentioned scenario, there are at least two different methods, which allow the exploitation of the advantages brought by scalable coding:

- Depending on the channel capacity, the DVB-H streams (HP and LP) can be merged together through a transcoder and sent directly to the DTMB terminal. In this way the overall bitrate is the sum of the base and enhancement layer, and the terminal will receive the high-quality video.

- The HP and LP streams are delivered over different TV channels. In this way, it is possible to reduce the overall bandwidth by simply using two different programs. In fact, the required bandwidth will be in any case much less than the simulcast scenario, where the streams are delivered at different quality levels. In our implementation, only the residual information is transmitted on the “high-quality” channel.
4.4 Pre-stored vs. live video transmission

In order to demonstrate the functionalities of the MING-T project in terms of video coding, two different scenarios with different complexity have been taken into account: i) the delivery of pre-stored video sequences, and ii) the live transmission. Both of them have important peculiarities and face different problems.

The most complex scenario, in terms of requested computational power and advanced coding features, is the encoding of pre-stored video sequences. In this scenario (where the real time constraint in the encoding process is not requested) one important feature is the exploitation of the motion analysis module. The motion analyzer is targeted at limiting the amount of data to be transmitted by reducing the frame rate of the video, therefore using the temporal scalability feature jointly with the FGS (on the LP stream).

The live scenario is instead the most demanding one in terms of computational power, because it requires the real-time delivery of videos. Therefore, there is no sufficient room for advanced coding features, at least within the performance characteristics of the current reference software. Following this idea, the most effective way to pursue the objectives is to push towards a simplified implementation of the scenario. This concept can be translated into the removal of the motion analysis feature and the consideration of only the FGS scalability (simple HP and LP streams), which allows increasing the coding speed up to 10-12 fps. This frame rate could be enough for video conferencing applications, while it is still too low for video broadcasting.

4.5 Motion analysis and scalability adaptation

The aim of the planned motion analysis module is to impose a further analysis on the video in order to reduce the amount of compressed data to be delivered. The motion analysis is carried out before compressing the video on the raw data. During the process, the most complex scenes (namely the time frames where the motion is high) are identified. These parts of the video are then coded in the most precise way (high frame rate), while in those time periods where the amount of motion is considerably low, a downsampling of the B-Frames can be performed. This provides a bitrate reserve to better encode the remaining frames. Note that the temporal downsampling is done on both streams (HP and LP); by constantly keeping the FGS information on the LP layer. The final solution for the motion analysis and adaptation module will evaluate the possibility of including spatially scalable videos as well, as in some particular transmission scenarios (e.g. sports) a higher frame-rate might be preferable over of a better (more detailed) visual quality. Final considerations about this aspect will be given in the project deliverable D3.6, where the final solution for scalable coding and adaptation will be described. So far, only the first preliminary studies regarding this topic have been performed.
5. EXPERIMENTAL RESULTS

In this chapter, a detailed description of the work carried out so far is reported. First, section 5.1 explains some general considerations and the initial issues we had to face. Next, the activities of the partners involved in the video coding part of the project are detailed. Section 5.2 describes and comments on the issues with the available rate control algorithm. Section 5.3 concentrates on the performance of the reference encoder on common test sequences, while section 5.4 explains possible solutions for unequal error protection schemes in DVB-H. Section 5.5 describes on the implemented RTP packetization scheme as proposed by the standardization body, while section 5.6 is targeted at describing the activities and tests performed on the decoder.

5.1 General considerations and work organization

So far, our development work has been concentrated on the accurate analysis of the reference software and its implemented features. During the analysis phase and the successive modifications to the codec, the following points need to be carefully analyzed.

- It is worth noting that the current version of the SVC codec reference software (standard repository of development team, actually working on version JSVM 8.9) is not able to perform real-time encoding and decoding. The implementation can encode a QVGA (320x240) sequence at a frame rate of around 3-4 fps (frames per second) in the best case, depending on the computational power of the encoding server. Therefore a strong effort must be spent in order to speed up the components.

- The decoder is faster, but still far from the real-time conditions requested by the project. In fact, the real-time constraint can probably be met on a high-performance machine, but in the final stage some modules must be added (and run simultaneously) to demodulate the signal and extract the real bitstream from the network packets. Therefore, it is questionable whether real-time decoding will be possible with the reference software on low-end devices like PDAs or smartphones.

- Because the complexity of decoding is much less than of encoding, the main challenge of optimizing the decoder is how to enhance the robustness of JSVM. No (wireless) channel can ensure that the transmission is perfect. Some packet-loss and bit-errors in the transmitted video bitstream are unavoidable. Unfortunately the available decoder (JSVM) can’t handle these transmission errors (and crashes easily). In order to get a usable SVC decoder, the robustness problem of decoder must be solved.

- In order to allow a fruitful collaboration among the partners cooperating on the video part, the overall implementation burden has been split between CN and THU. In this way it will be possible to concentrate the efforts on the encoder and decoder part respectively. Since the development of the codec structure must follow strict rules in terms of interfaces (between the encoder part and the decoder part) and common synchronization points (headers and timestamps), a document has been already prepared for internal circulation in order to establish a common communication interface.

- The available software (both encoder and decoder) only implements the Annex B format, requiring the full implementation of the RTP format. Since the documentation is still poor, [13] has been adopted as the reference document, being confident that the final version of the packetization format won’t be too far from it.
5.2 Encoder rate control

The reference software encoder generates a proof of concept stream, containing all layers (base and enhancements all together) that can then be extracted by running an appropriate tool called BitStreamExtractor. This allows extracting a selected layer at the desired quality. Unfortunately the mechanism is not suitable for the transmission over the a broadcast channel; in fact in this case two separate streams must be obtained and delivered on the High and Low Priority Link and there is no a priori knowledge about the channel conditions of each user. Therefore the first task was to efficiently structure the bitstream, in order to flexibly extract the base layer and an FGS enhancement layer with a close look at the rate control algorithm. One of the main problem we have encountered is that the available rate control module is not very efficient and it is more suitable for video storage instead of video streaming. Ongoing tests are targeted at correcting the module behavior. The irregular rate control can be noticed from Figure 15, where the algorithm (tested on the well-known Foreman test sequence with a GOP length of 4 frames, suitable for streaming) is extremely unstable in the time frames where the motion (camera or objects) are high. In each test, the base layer has been set to 200kbps and the enhancement layer at 400kbps.

The same test has been run varying the GOP size. As it can be noticed by Figure 14, Figure 15, Figure 16 and Figure 17, the algorithm is more stable as we reduce the GOP size, while it diverges heavily when the GOP size increases.
In the graphs, the single points have been derived calculating the effective bit-rate on a one-second basis. As it can be noticed, the base layer is quite unstable except when the GOP is very short and the achieved results are not satisfactory if analyzed globally. The rate is correct if calculated over the whole sequence, but it fluctuates strongly if analyzed on a small time window. In fact, when high peaks occur, they are usually compensated within the next 50 or 75 frames. The pink curve reflects the behaviour of the FGS enhancement. Even in this case fluctuations are very strong. The yellow line represents our rectification of the FGS bandwidth assignment according to the base/enhancement ratio imposed by the hierarchical modulation in DVB-H. Although the best results in terms of rate control reliability are given by setting the GOP size =1, the system won’t be able to often achieve a good quality in this case, and therefore tests have been done using a GOP length of 4 frames.

5.3 Encoder performance on example sequences

Figure 18 and Figure 19 represent the performance in terms of encoding speed obtained when varying the motion search range and the spatial resolution of the video. A small value can significantly improve the encoder performance, although this remains still far below the real-time condition. The speed is calculated only on the base layer. When applying the scalable coder, the speed is reduced approximately by 11% depending on the resolution.

Tests are performed using the “foreman” video sequence at different spatial resolutions. The chosen GOP size is 4 with an Intra refresh of 64 frames.
A similar test has been conducted in order to evaluate the performances varying the GOP size from 1 (no temporal scalability) to 64 on the QVGA sequence. Figure 20 demonstrates the increasing complexity with higher values for the GOP size due to the high number of dyadic predictions involved.

The numbers in Table 2 present instead the general performances of the codec in terms of Peak-Signal-To-Noise-Ratio (PSNR) and the achievable gain in quality by transmitting the FGS layer on the Low Priority link.
Table 2. Quality enhancement provided by the FGS layer at double rate with respect to the base layer.

<table>
<thead>
<tr>
<th>Sequence and size</th>
<th>Base rate [kbps]</th>
<th>Enh rate [kbps]</th>
<th>Base PSNR [dB]</th>
<th>Enh PSNR [dB]</th>
</tr>
</thead>
<tbody>
<tr>
<td>foreman 320x240</td>
<td>52</td>
<td>103</td>
<td>28.21</td>
<td>29.92</td>
</tr>
<tr>
<td>foreman 320x240</td>
<td>105</td>
<td>202</td>
<td>32.46</td>
<td>34.19</td>
</tr>
<tr>
<td>foreman 320x240</td>
<td>204</td>
<td>403</td>
<td>36.26</td>
<td>38.33</td>
</tr>
<tr>
<td>foreman 352x288</td>
<td>55</td>
<td>104</td>
<td>26.82</td>
<td>28.26</td>
</tr>
<tr>
<td>foreman 352x288</td>
<td>104</td>
<td>202</td>
<td>30.88</td>
<td>32.42</td>
</tr>
<tr>
<td>foreman 352x288</td>
<td>200</td>
<td>401</td>
<td>34.48</td>
<td>36.11</td>
</tr>
<tr>
<td>mobile 320x240</td>
<td>108</td>
<td>218</td>
<td>24.35</td>
<td>25.68</td>
</tr>
<tr>
<td>mobile 320x240</td>
<td>211</td>
<td>422</td>
<td>28.64</td>
<td>30.22</td>
</tr>
<tr>
<td>mobile 320x240</td>
<td>413</td>
<td>843</td>
<td>32.11</td>
<td>34.34</td>
</tr>
<tr>
<td>mobile 352x288</td>
<td>103</td>
<td>213</td>
<td>19.86</td>
<td>21.43</td>
</tr>
<tr>
<td>mobile 352x288</td>
<td>212</td>
<td>431</td>
<td>26.52</td>
<td>27.84</td>
</tr>
<tr>
<td>mobile 352x288</td>
<td>409</td>
<td>819</td>
<td>29.49</td>
<td>31.19</td>
</tr>
</tbody>
</table>

As can be seen from the numerical results, the advantage of introducing a quality enhancement is not particularly evident, as the addition of the high frequency information (the FGS layer) is very demanding in terms of bitrate. In the reported results, in fact a double rate for the enhancement layer has been considered with respect to the base layer. This assumption is not the only possible choice, because in this case we have considered the same modulation code rate, meaning that the both streams are protected in the same way.

5.4 Encoding for un-equal error protection

According to the concepts of unequal error protection, we could slightly modify this limitation and assign a lower protection to the Low Priority Stream, adopting a different code-rate ratio. The base layer instead requires a higher protection, as its correct reception is mandatory.

The available possibilities for the code-rate in the DVB-H standard are: 1/2, 2/3, 3/4, 5/6 and 7/8. When setting a lower code-rate for the enhancement layer, a larger file can be sent, allowing for a better quality when the receiving conditions are good. Therefore we have decided to assign a code rate of 1/2 to the base layer and 3/4 to the enhancement layer for a second set of experiments. In this way the ratio between the LP and HP stream can be around 3 instead of 2.

In order to validate the effectiveness of the selected configuration, we have tested the encoder on a set of real video sequences. The tests (reported in Table 3) show that the achieved quality gain with respect to the single layer transmission can be assessed around 3-4 dB. This configuration has been tested on three different sequences, directly captured from a DVB-T grabber (MPEG-2 format) that represent three common video scenarios:

- A sport sequence
- A cartoon
- A sit-com
Table 3. Quality and rate control tests on real TV sequences.

<table>
<thead>
<tr>
<th></th>
<th>HP layer</th>
<th>LP layer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sport</td>
<td>150kbps</td>
<td>151kbps</td>
</tr>
<tr>
<td>Cartoon</td>
<td>150kbps</td>
<td>151kbps</td>
</tr>
<tr>
<td>Sit-com</td>
<td>150kbps</td>
<td>151kbps</td>
</tr>
</tbody>
</table>

The tests performed on those video sequences demonstrate that the encoder can achieve good performance (with a HP/LP bit-rate ratio of 1:3), even though the final bitrates tend to strongly deviate from the real desired level, especially in the enhancement layer. Our tests also prove that the selected GOP length and the Intra-refresh can suite the quality requirements in most situations as the base layer already achieves an average PSNR higher than 30 dB. With such a low bandwidth (150 kbps for the base layer), some artefacts are introduced when there is a sudden scene change right after an intra frame. These blocking artefacts are usually compensated when the enhancement layer is decoded. In figure 21 samples of the three test sequences are reported.
Figure 21. Typical frame quality for the base and enhancement layer of the sport sequence (a) (b), the cartoon (c) (d) and the sit-com (e) (f). Base quality and enhancement improvements for sport (g) (h), cartoon (i) (j) and sit-com (k) (l). Base-layer bitrate is 150 kbps, and enhancement layer bitrate is 450 kbps.

5.5 Stream-format and RTP-packetizing

Before specifying in detail the features already implemented, let us have a quick look at the main objective of this procedure. In order to correctly deliver the stream, the video is divided into so-called Access Units (NALUs) each containing a portion of the visual content. Each NALU can carry a maximum payload corresponding to one frame of the video sequence. In practice however, this is often infeasible as (for example) intra frames are typically bigger than the maximum transport unit size (MTU), which is set to 1500 bytes.

Therefore fragmentation or aggregation (in case frames are too small) rules must be developed accordingly. The available fragmentation and aggregation rules are described in detail in [12] and [13] and therefore only the strategy implemented by us will be discussed here. Taking the RTP format of AVC as an example, we have initially implemented the Single NALU packetization. As already mentioned above, this method is not a viable solution for the aim of our project, because it does not efficiently utilize the available bandwidth. Therefore the aggregation MTAP16 (Multi-Time aggregation Packet 16 bits) and fragmentation FU-B (Fragmentation Unit Type B) rules have been applied also. In the following some details about the implementation will be given.

The actual header information for each frame is set according to the AnnexB format, and therefore the start code is a 4 byte word:

0x00000001

After these 4 bytes the NALU header starts. It can be bigger than 1 byte, but the first byte is the most important. The structure of the NALU header in this simple case is [RFC3984][12]:

+---------------------------+
### F: 1 bit

forbidden_zero_bit. The H.264 specification declares a value of 1 as a syntax violation.

<table>
<thead>
<tr>
<th>F</th>
<th>NRI</th>
<th>Type</th>
<th>PAYLOAD</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### NRI: 2 bits

nal_ref_idc. A value of 00 indicates that the content of the NAL unit is not used to reconstruct reference pictures for inter picture prediction. Such NAL units can be discarded without risking the integrity of the reference pictures. Values greater than 00 indicate that the decoding of the NAL unit is required to maintain the integrity of the reference pictures.

### Type: 5 bits

nal_unit_type. This component specifies the NAL unit payload type

---

**In order to implement a file format for streaming over the Internet, the 1 byte header must be replaced by the UDP [RFC 1122] and RTP [RFC 3550] headers.**

In the RTP stream format, each packet needs to implement an UDP and an RTP header, described as follows.

**UDP header:**

```
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
+---+---+---+---+---+---+---+---+
|    |    |    |    |    |    |    |    |
+---+---+---+---+---+---+---+---+
|          SOURCE PORT          |       DESTINATION PORT        |
+---+---+---+---+---+---+---+---+
|             LENGTH            |           CHECKSUM            |
+---+---+---+---+---+---+---+---+
| ++++++++++++++++++++++++++++++++ | ++++++++++++++++++++++++++++++++ |
|     | SOURCE PORT                 | DESTINATION PORT              |
| +-----+--------------------------+------------------------------+
|      | LENGTH                      | CHECKSUM                      |
| +-----------------------------+--------------------------------+
```

**Source Port.** 16 bits.
The port number of the sender. Cleared to zero if not used. In fact, at the moment we do not have any specification about the ports that will be used in the final application.

**Destination Port.** 16 bits.
The port this packet is addressed to. (Same considerations as above about the actual value)

**Length.** 16 bits.
The length in bytes of the UDP header and the encapsulated data. The minimum value for this field is 8.

**Checksum.** 16 bits.
Computed as the 16-bit one's complement of the one's complement sum of a pseudo header of information from the IP header, the UDP header, and the data, padded as needed with zero bytes at the end to make a multiple of two bytes. If the checksum is cleared to zero, then checksumming is disabled. If the computed checksum is zero, then this field must be set to 0xFFFF.
RTP header:

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | 1 |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | 1 |
| V=2 | P | X | CC | M | PT | sequence number |
| +++++++-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| +++++++-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| +++++++-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| +++++++-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

The RTP header information to be set according to this RTP payload format is set as follows:

**Marker bit (M):** 1 bit
Set for the very last packet of the access unit indicated by the RTP timestamp, in line with the normal use of the M bit in video formats, to allow an efficient playout buffer handling. For aggregation packets (STAP and MTAP), the marker bit in the RTP header must be set to the value that the marker bit of the last NAL unit of the aggregation packet would have had, if it were transported in its own RTP packet. Decoders may use this bit as an early indication of the last packet of an access unit, but must not rely on this property.

**Payload type (PT):** 7 bits
The assignment of an RTP payload type for this new packet format is outside the scope of this document and will not be specified here. The assignment of a payload type has to be performed either through the profile used or in a dynamic way.

**Sequence number (SN):** 16 bits
Set and used in accordance with RFC 3550. For the single NALU and non-interleaved packetization mode, the sequence number is used to determine decoding order for the NALU.

**Timestamp:** 32 bits
The RTP timestamp is set to the sampling timestamp of the content. A 90 kHz clock rate must be used.

**Synchronization source:** 32 bits
They are set to 0x12345678, the same value of RTP packetization in H.264 AVC.

In our transmission the first 16 bits are:

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
| V=2 | P | X | CC | M | PT |
| +++++++-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| 1 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 |

According to the definition of the RTP header, in our implementation one field has been discarded, as its length depends on the value of the CC field (set to 0 in our case). This field is called “Contributing Source (CSRC) Identifiers”.

After the insertion of the UDP and RTP headers the NALU header begins. The NALU packets can be classified into three different categories:
- Single NAL Unit Packet
- Aggregation Packets
- Fragmentation Units

5.5.1 Single NAL Unit Packet

The single NAL unit packet defined here must contain only one NAL unit. This means that neither an aggregation packet nor a fragmentation unit can be used within a single NAL unit packet. A NAL unit stream composed by decapsulating single NAL unit packets in RTP sequence number order must conform to the NAL unit decoding order. The structure of the single NAL unit packet is shown in the next figure:

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|F|NRI|  type   |                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|               Bytes 2..n of a Single NAL unit                 |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Unfortunately, this mode cannot be used as our video (scalable) needs to be transmitted over two separate RTP sessions. Therefore, the single packetization (non interleaved packetization) cannot be used and the interleaved mode must be adopted.

5.5.2 Aggregation Packets

RFC3984 specifies that some aggregation formats can not be used when using the interleaved mode. In particular, only STAP-Bs and MTAP-Bs may be used.

During the work done so far, we have decided to adopt the MTAPs format. In particular, we went for the MTAP16. In the following scheme, an example of aggregation for two different NALUs is provided.
Citing the RFC: “The timestamp offset field must be set to a value equal to the value of the following formula: If the NALU-time is larger than or equal to the RTP timestamp of the packet, then the timestamp offset equals (the NALU-time of the NAL unit - the RTP timestamp of the packet). If the NALU-time is smaller than the RTP timestamp of the packet, then the timestamp offset is equal to the NALU-time + (2^32 - the RTP timestamp of the packet)”. The NALU x HDR corresponds to the standard NALU header where types are specified in section 5.4, table 3.

Further information about the format can be directly found in the RFC itself.

5.5.3 Fragmentation Units (FUs)

This payload type allows fragmenting a NAL unit into several RTP packets.

Fragmentation is defined only for a single NAL unit and not for any aggregation packets. A fragment of a NAL unit consists of an integer number of consecutive octets of that NAL unit. Each octet of the NAL unit must be part of exactly one fragment of that NAL unit. Fragments of the same NAL unit must be sent in consecutive order with ascending RTP sequence numbers (with no other RTP packets within the same RTP packet stream being sent between the first and last fragment). Similarly, a NAL unit must be reassembled in RTP sequence number order.

When a NAL unit is fragmented and conveyed within fragmentation units (FUs), it is referred to as a fragmented NAL unit. STAPs and MTAPs must not be fragmented. FUs must not be nested; i.e., an FU must not contain another FU. The RTP timestamp of an RTP packet carrying an FU is set to the NALU time of the fragmented NAL unit.

The next figure presents the RTP payload format for FU-Bs. An FU-B consists of a fragmentation unit indicator of one octet, a fragmentation unit header of one octet, the DON code and a fragmentation unit payload. Note that the RTP padding is not necessary, as all data are byte aligned.
The NAL unit type FU-B must be used in the interleaved packetization mode for the first fragmentation unit of a fragmented NAL unit, while NAL unit type FU-B must not be used in any other case. In other words, in the interleaved packetization mode, each NALU that is fragmented has an FU-B as the first fragment, followed by one or more FU-A fragments.

The FU indicator octet has the following format:

\[ \begin{array}{c}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
\hline
\text{FU indicator} & \text{FU header} & \text{DON} \\
\end{array} \]

\[ \begin{array}{c}
\text{FU payload} \\
\vdots \text{OPTIONAL RTP padding} \\
\end{array} \]

The value equal to 29 in the Type field of the FU indicator octet identify an FU-B. The value of the NRI field must be set according to the value of the NRI field in the fragmented NAL unit.

The FU header has the following format:

\[ \begin{array}{c}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
\hline
\text{S} & \text{E} & \text{R} & \text{Type} \\
\end{array} \]

\text{S:} 1 bit
When set to one, the Start bit indicates the start of a fragmented NAL unit. When the following FU payload is not the start of a fragmented NAL unit payload, the Start bit is set to zero.

\text{E:} 1 bit
When set to one, the End bit indicates the end of a fragmented NAL unit, i.e., the last byte of the payload is also the last byte of the fragmented NAL unit. When the following FU payload is not the last fragment of a fragmented NAL unit, the End bit is set to zero.

\text{R:} 1 bit
The Reserved bit must be equal to 0 and must be ignored by the receiver.

\text{Type:} 5 bits
This number refers to the NALU fragmentation method, which in our case is 29. The definition of these codes can be found in Section 5.4 of RFC-3984.
The FU payload consists of fragments of the payload of the fragmented NAL unit so that if the fragmentation unit payloads of consecutive FUs are sequentially concatenated, the payload of the fragmented NAL unit can be reconstructed. The NAL unit type octet of the fragmented NAL unit is not included as such in the fragmentation unit payload, but rather the information of the NAL unit type octet of the fragmented NAL unit is conveyed in the F and NRI fields of the FU indicator octet of the fragmentation unit and in the type field of the FU header. An FU payload may have any number of octets and may be empty. If a fragmentation unit is lost, the receiver should discard all following fragmentation units in transmission order corresponding to the same fragmented NAL unit.

For convenience the RTP timestamp has been set in our implementation according to the decoding order. The initial header of the SVC file will have its timestamp set to 0 (zero) as this does not refer to any specific frame. The first frame will then have its timestamp set to 0 again. In a similar way to AVC, the time increment is set to 1000. Therefore the second coded frame will have a value of 1000 as its timestamp and so on. At the same time, all NALUs corresponding to the same frame will have the same timestamp.

5.6 Decoder

This section describes our concept to decode the scalable-bitstream generated by the reference encoder and packetized according to the formats detailed in the previous section. First, section 5.6.1 explains how to reconstruct a suitable scalable video bitstream from the received incoming data packets, while section 5.6.2 concentrates on the synchronization of video-frames distributed by separate parallel RTP streams. Finally, section 5.6.3 describes first results regarding the performance of our decoder.

5.6.1 SVC Stream Extraction

The NAL units of SVC streams are classified into VCL NAL units, which contain coded pictures or coded picture data partitions, and non-VCL NAL units, which contain associated additional information. The most important non-VCL NAL units are parameter sets and Supplemental Enhancement Information (SEI). The sequence and picture parameter sets contain infrequently changing information for a video sequence. They will be referenced by VCL packets in the same picture or sequence. The reference relation of the parameter sets and slice packets is shown in Figure 22.

![Figure 22. Reference relation of SVC stream packets](image-url)
Because of that the SVC Extractor will extract a sub-stream from the input SVC stream, and the new sub-stream will need a new set of decoding parameters. The first task of the SVC Extractor is to generate the new non-VCL NAL units of the output sub-stream. The reference of the new parameter set generation process are the requirements of the terminal decoder and the parameter sets of the input SVC stream. The other task of the SVC Extractor is to filter the VCL NAL packets of different SVC video layers, i.e. to keep the wanted layer’s packets and discard the unwanted layer’s packets. Figure 23 is the initial designed structure of our SVC Extractor.

Figure 24. Extractor structure.

Different scalability features of the SVC streams have different filter and extraction principles. For temporal scalability, the performance test of the SVC decoder shows that the GOP size of the SVC streams is tied to the decoding complexity of the corresponding SVC stream. Discarding some temporal scalable layers’ NAL units could effectively decrease the GOP size. The main factors that affect the temporal scalable layers’ NAL units filter are the power restrictions and the computing performance of the terminal. For the spatial scalability filter, the main factor that affects the selection of spatial scalable layers is the screen resolution of the terminal. According to our test in section 5.6.3, the decoding complexity and buffer consumption of the decoder decreases significantly with the spatial resolution of the video stream. Of course, the terminal’s power restriction, its computing performance and the available memory are also important factors of the spatial scalable filter process.

For the quality scalability filter process, some packet loss can be expected in the enhancement stream, because we transmit the quality scalable layers over a weakly protected channel. The distortion due to these packet losses is unavoidable in a broadcast system over a wireless channel. Unfortunately, the current version of the JSVM decoder is not optimized for robustness, and will often crash when encountering input sequences with missing or incomplete packets. In order to get a robust system, this problem must be resolved. In our preliminary implementation of the terminal software, the SVC extractor analyzes the packet loss condition of the current stream and only extracts the lowest whole quality layer in the stream and sets this layer as the target extracting layer of the filter process of the SVC Extractor. So the main factor that affects the quality scalable layer extraction is the packet loss rate on the terminal side.

To summarize, the first main task of the SVC extractor is to generate new stream parameter sets. And the other tasks of the extractor are to filter these slice packets: to keep the wanted layer’s packets and discard the unwanted layer’s packets.
5.6.2 Cross-Layer Synchronization by using RTP

SVC, as all previous video compression standards, requires the syntactical entities of the bit stream to be presented to the decoder in a certain order, the decoding order. In the case of SVC, the decoding order is expressed in constraints for the sequencing of the NAL units. Some SVC profiles allow a certain amount of NAL unit reordering without breaking compliance, but others do not. Because we need to transmit the base layer and enhancement layers in LP and HP respectively, we must embed the base layer and the enhancement layers in two different RTP sessions. In this case, the situation gets even more complicated. Fortunately, the RTP standard offers a cross-layer synchronization mechanism that can be used for our SVC transport scenario.

Figure 25 illustrates an example of the cross-layer synchronization by using RTP. The base layer (layer 0 in the figure) is a QCIF@15 Hz bit stream, and the quality is enhanced by an SNR scalable layer (layer 1) with the same frame rate. Based on the SNR scalable layer, a spatial enhancement layer (layer 2) of CIF@30 Hz is encoded, and finally an MGS layer (layer 3) is encoded based on the spatial enhancement layer. Three access units are depicted in the figure. Layer 0 is transported in RTP session S1, using STAP-B. Layer 1 is transported in RTP session S2, with the NAL unit in access unit 1 being fragmented into two FUs (FU-B with the first packet and FU-A with following packets). Layers 2 and 3 are transported in the same RTP session S3, using both STAP-B and MTAP. As can be seen, the DON values indicate the decoding order of all NAL units across all the layers. This way, the receivers can easily recover the NAL unit decoding order from the signaled or derived DON values. For a receiver that receives only the base layer, there will be gaps in the DON values of some NAL units, which is compliant with RFC3984. However, and fortunately, the derived decoding order would still be correct.

5.6.3 Performance Test of the JSVM Decoder

Figure 26 and Figure 27 represent the performance in terms of decoding speed obtained varying the motion search range and the spatial resolution of the video\(^2\). The decoding speed is very close to the real-time requirement (on a current PC), and the search range has little effect on the decoding performance. The speed is calculated only on the base layer. When applying the

\(^2\) Tests are performed using the “foreman” video sequence at different spatial resolutions. The chosen GOP size is 4 with an Intra refresh of 64 frames.
scalable coder, the speed is reduced approximately by 39% depending on the resolution, for QVGA about 17 fps.

![Graph showing decoding speed with varying search range](image)

**Figure 26. Decoding Speed for the foreman sequence (QVGA) changing the motion search range.**

![Graph showing decoding performance with varying spatial resolution](image)

**Figure 27. Decoding Speed for the foreman sequence changing the spatial resolution.**

A similar test has been conducted in order to evaluate the performance when varying the GOP size from 1 (no temporal scalability) to 64 on the QVGA sequence. Figure 27 demonstrates that decoder framerate drops only slightly despite the increasing complexity implied by the larger GOP size.
Figure 28. Decoding Speed at variable GOP size.
6. NEXT STEPS

According to the simulations results achieved so far, including the implemented routines for embedding the stream into the RTP payload format, the activity on the scalable encoder and decoder will concentrate on the following aspects:

- Rate-control adaptation according to the constraints imposed by time-slicing and available bit-rate
- Implementation of the motion analyzer module for offline and pre-stored video sequences to appropriately select the best scalability configuration
- Improvement of the execution speed for the online encoder and for the decoder
- Implementation of the RTP format on the decoder
- Implementation of the acquisition module to capture live streams
- The possibility of introducing the spatial scalability as an additional feature to make the system even more flexible.
- Enhancement of the decoder robustness

Parallel to our implementation work of the MING-T scalable video solution during the last months, the JVT team has worked towards the standardization of the SVC codec. Very recently, a special issue about SVC has been published on the IEEE Transactions for Circuits & Systems for Video Technology (vol. 17, n. 9), summarizing the latest information about the codec and the standardization progress. In particular, the RTP file format has been fixed, and the final specification is compliant with the implementation we have carried out so far. On the other hand, the scalability features of SVC have been modified slightly. The FGS scalability has been replaced by an improved mechanism of so-called Medium Grain Scalability (MGS), which promises to perform better both in terms of computational complexity and resulting overhead. Please check the bibliography for the references to the recent TCSVT papers.

Unfortunately, it was impossible to cover the new codec options in this document, because MGS is a very recent development and the available time was not sufficient to produce meaningful results. Of course, the new scalability features will replace the FGS and will be included in our upcoming report D3.5 which will describe the final solution for the scalable video coding within the MING-T project.


