Video adaptation based on the SVC file format

Eduardo Martínez Graciá
Jordi Ortiz Murillo
Rafael López Pérez
Antonio F. Skarmeta
Intelligent Systems Group, University of Murcia, Spain
Email: edumart@um.es, jordi.ortiz@um.es, rafalp@um.es, skarmeta@um.es

***Abstract*—This paper provides a description of the Scalable Video Coding (SVC) file format standard and its support for the video adaptation in a streaming scenario. The paper shows the advantages of using SVC and its standard file format when implementing server driven and receiver driven adaptation. The organization of information inside the SVC file permits the efficient application of adaptation procedures based on filtering at the server or at media aware network elements, carrying out receiver driven layered multicast and multi-interface transmission. The paper provides an overall description of a tool that supports the creation of files compliant with the SVC file format.**

***Keywords*-Scalable video coding, SVC file format, streaming, video adaptation, multicast and multi-interface transmission.**

I. INTRODUCTION

This paper introduces the reader to the ISO media file format [1] and its SVC extension [2] as well as its usage to stream SVC videos over the network in an adaptable way. We expose how to carry out the adaptation executed by the sender or the receiver and describe a tool to ease this process.

The aim of this paper is not introducing the reader to the SVC video codec or the protocols involved in a streaming service. We point the reader to [3] and [4] for a more detailed introduction of those technologies.

SVC is the scalable extension of H.264/AVC. It provides three scalability dimensions: temporal, allowing diverse frame rates; spatial, relative to the picture size; and quality, allowing different values of the signal-noise ratio. In SVC, each feasible combination of these three characteristics is called *a layer*. A layer is recognized by a group of three identifiers (ID): Dependency ID (DID), Quality ID (QID), and Temporal ID (TID). All the layers up to some specific values of these identifiers determine an *operation point* of the SVC stream, that is, a subset of the bitstream that can be decoded to generate a representation of the video with a certain quality. This group of three identifiers will be referred to as DQT from now on in this paper.

As an extension to AVC, SVC defines the Video Coding Layer (VCL) and the Network Abstraction Layer (NAL). VCL is intended to optimally represent video data while NAL provides efficient data formatting to facilitate video delivery or storage. NAL defines the minimum AVC data association, called NAL Unit (NALU). A NALU starts with one octet long header followed by raw byte sequence payload. SVC NALUs extend their header with three more octets, as can be seen in Figure 1. These new fields identify the layer associated to the NALU (DID,QID,TID). The NALU header co-serves also as RTP payload header [5].

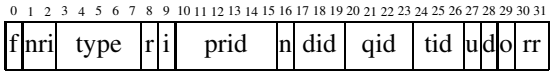

Figure 1. AVC NAL Unit header followed by its SVC extension

The SVC file format specification defines how SVC streams are stored in any file container based on the ISO Base Media file format [1] (e.g., MP4 files [6]). It is a specialization of the AVC file format [7] that incorporates some new structures to enable SVC specific features, including *adaptation operations* and *erosion storage*. Adaptation operations consist in discarding SVC data, according to an adaptation decision based on the context in which the file is used, whereas erosion storage is the reduction of SVC storage space with a controlled video quality degradation. A recommended reading about the SVC file format can be found in [8].

In order to provide a representation of the video specified by a certain operation point, an adaptation service is used. The service removes any SVC NALU over the desired operation point simply looking at their SVC NALU extension header. Taking into account the location where the adaptation decision is made, there are two possible strategies to perform the adaptation of the SVC video: sender driven adaptation and receiver driven adaptation.

Sender driven adaptation is interesting when using unicast streaming (video on demand). The adaptation is done at the streaming server itself or at a media aware network element (MANE) [9] in charge of thinning the bitstream. The operation point is selected by an integrated function of the server or MANE, or in a separate module responsible of the decision logic.

Receiver driven adaptation is more appropriate when using multicast or multi-interface reception. In these cases, the client is the entity in charge of selecting the best operation

 ISBN: 978-1-61208-275-2

point. When multicast is used, the adaptation is performed by means of the well known receiver driver layered multicast [10]. On the other hand, if multi-interface reception is available, the receiver starts as many streaming sessions as needed, with different interface endpoints in each session. The whole set of active streaming sessions deliver the layers of the selected operation point.

The remainder of this paper presents the file management procedures that support the sender and receiver driven adaptation. Section II presents an overview of the MP4 file format and the specialization of this file format to store SVC bitstreams; section III describes the storage of SVC bitstreams into tracks in MP4 files and the creation of extractor and hint tracks. Section IV describes the use of SVC files for sender driven adaptation, and section V presents the use in receiver driven adaptation. Section VI introduces a tool and show some results. Finally, in section VII we conclude this paper and provide future work plans.

II. File format

A. Overview of MP4 file format

The MP4 file format is an extension of the ISO Base Media file format designed to store MPEG-4 multimedia presentations. This standard defines a generic and codec agnostic multimedia container based on an object oriented design. Files are made up of objects, called *boxes* or *atoms*, and every piece of data in the file is stored in a box. Inheritance and composition relationships are commonly used in the definition of boxes.

The standard defines three coordinated arrangements of information: a logical structure, a time structure, and a physical structure. The logical structure reflects the components of a movie, that is, a set of time-parallel tracks. The time structure represents the sequence of samples of each track, according to the timeline of the overall movie. And the physical structure of the file defines the location of the information, including the boxes that contains the logical and time structure description, and the media data samples themselves. An interesting aspect of this file format is the separation of the media data from the metadata. For example, the length and timestamp of a video frame are not stored adjacent to the frame itself, but in boxes in the metadata part. Additionally, to facilitate editing, timing data is relative and media data is stored as it is, without any kind of transformation. When data must be streamed, special instructions are stored in hint tracks to support servers in the process of data packetization.

The main top level boxes of the MP4 file format are the movie atom *moov*, which is the metadata container, and the media data atom *mdat*, which is the data container. Inside the movie atom there is a movie header atom *mvhd* and a set of *trak* atoms. The movie header stores media independent information relative to the whole media. Regarding the *trak* atoms, each of them contains the metadata related to a time series of media. The most important boxes included inside *trak* are:

- A track header atom *tkhd* that contains the track identifier and duration.
- A track reference atom *tref* that contains the identifiers of other tracks with which this one has some type of dependency (‘hint’ and ‘scal’ are examples).
- A media atom *mdia*, explained below.

The media atom includes several boxes that permit storing efficiently the information of a media stream. First of all, the handler box *hdlr* specifies the nature (audio, video, hint, etc.) of the media in the track. Additionally, the sample table box *stbl* contains all the information to obtain the physical location and timing of each sample. This information is structured in a set of boxes that are formatted as compact tables.

The compact representation of the physical location is based on the observation that groups of frames from the same track are often stored contiguously in the *mdat* box, even when multiplexing various tracks. The concept of *chunk* of samples represents this grouping of contiguous samples from the same track. Several levels of indirection are used to locate a sample: the sample to chunk box *stsc* permits to find the chunk that contains a sample; then, the chunk offset box *stco* specifies the offset of each chunk from the beginning of the file; finally, the sample size box *stsz* stores the size of each sample, and therefore, using the chunk offset and the sizes of the preceding samples in the chunk, it is possible to find the offset of the sample from the beginning of the file.

Regarding the timing information of a track, the sample table box contains two tables that provide a relation between decoding time and sample number *stts*, and composition time with sample number *ctts*. The decoding time of a sample is stored as its duration (or delta from the preceding sample) to obtain a compact representation, whereas the composition time is stored as a time offset from the decoding time of the sample.

We have previously mentioned that the MP4 file format is codec agnostic. To facilitate the storage of different media types, there are track specializations for each media nature parameterized by the *description of sample entries*. The sample description box *stsd* included in sample table box gives detailed information about the coding type used, and any initialization information needed to configure the decoder. The standard enables the use of several sample descriptions per track, although usually there is only one. Each chunk has an index that specifies the sample description of all the samples included in the chunk. The AVC and SVC file formats are not, strictly speaking, truly file formats on their own, but specifications of new sample descriptions and additional boxes to describe specific metadata for the AVC and SVC codecs. We will go over these additions in following sections.

 ISBN: 978-1-61208-275-2

When media is delivered using a streaming protocol, the characteristics of the transmission system may force to reformat the media data, stored in its natural form, in order to perform the communication. This is especially evident when the transmission is done on a packet based network like Internet. One of the particularities of Internet is the Maximum Transfer Unit (MTU) that establishes the maximum packet size that can be transmitted without fragmentation from a particular source to a particular destination. If this value was not taken into account during the media production, some aggregation or fragmentation would be needed to fit the media chunks into packets. The RTP protocol [11], the universal protocol for real-time media transmission, was designed in an extensible manner forcing the definition of different payload format or profile documents for each media to be transported, e.g., H.264/AVC Payload format [9]. This document defines the way media data has to be fragmented and signalled when transported over RTP.

A streaming server may calculate the packetization at streaming-time. However, the MP4 file format defines special tracks that help during the packetization process. So called *hint* tracks contain instructions to assist media-independent streaming servers in the generation of packets. Such instructions contain references to sample data fragments from one or more media tracks, listed as hint-type references in the *tref* box of the hint track. Special authoring tools called hinters have the payload format specific knowledge required to generate hint tracks after the editing process of the media tracks is completed. Hint tracks are structured like media tracks, and as a result, a file may contain both the original media and one or more hint tracks referencing the media track. Each hint track can support just one streaming protocol specified in the sample description of the hint track.

Streaming servers using RTSP [12] usually employ SDP [13] as the description format of the presentation. The MP4 file format defines a set of boxes to store SDP information during the hinting process, to assist the server in forming a full SDP description. SDP boxes are stored at both the movie and the track level. The movie SDP box contains the session level description, whereas track SDP boxes keep the lines corresponding to media level descriptions.

B. Overview of the SVC file format

The SVC file format is designed to be back compliant with the AVC file format, and consequently we start with a quick review of the latter.

The AVC file format defines extensions for the ISO Base Media file format to cope with three interesting features of the AVC codec: switching pictures, sub-sequences and parameter sets. Switching pictures and sub-sequences are AVC mechanisms for video adaptation that have been superseded by SVC, so they will not be addressed in this paper. Regarding parameter sets, the AVC codec defines sequence and picture parameter sets to convey infrequently changing decoder configuration. These parameters are usually transmitted out of band to the receiver, and more specifically, inside the SDP presentation description. To support the decoupling of parameter sets from coded video data, the AVC file format permits two possible mechanisms to store AVC data in tracks:

- A single video elementary stream: in this case a unique track contains samples that store video coding related NAL units; the track may also store non-video coding related NAL units such as SEI messages and access unit delimiter NAL units. Sequence and parameter set NAL units are stored in the sample description box of the track (*stsd*).
- Video and parameter set elementary streams: in this case one track contains samples that store the same type of NAL units indicated in the previous option, but sequence and parameter set NAL units are not stored in the sample description of this track; instead of that, they are stored as data samples of a second track. This option may be interesting when there are unusual frequent changes in the parameter sets of the video stream.

Samples in AVC tracks correspond to access units in AVC terminology, that is, complete video pictures. Each access unit is made up of a set of NAL units, that usually contain complete picture slices. To identify the limits between NAL units of the same sample, each NAL unit is preceded with a field that specifies its length in bytes. The length field can be of 1, 2, or 4 bytes. The definition of AVC samples specified in [7] is:

```
aligned(8) class AVCSample {
   unsigned int pictureLength = sample_size;
   for (i=0; i<pictureLength; ) {
      unsigned int((LSMO+1)*8) NALUnitLength;
      bit(NALUnitLength * 8) NALUnit;
      i += (LSMO+1) + NALUnitLength;
   }
}
//LSMO stands for Length Size Minus One
```

A video elementary stream is stored in a track identified by means of a sample description that has the coding name ‘avc1’. Sample descriptions of this type have a special box called *avcC* that contains the AVC decoder configuration record. This record includes the list of sequence and parameter sets, specifies the AVC profile and level, and the amount of bytes (LSMO) used to represent the NAL units length. On the other hand, tracks that store parameter sets have the coding name ‘avcp’.

After this short description of the main features of the AVC file format, we can continue the discussion of the SVC file format. In addition to a specific sample description, there are three special constructions defined in the SVC file format to support scalability: *extractors*, *tiers* and *temporal metadata statements*.

Extractors are the first and simplest instrument to support

 ISBN: 978-1-61208-275-2

SVC adaptation. An extractor contains special NAL units (with type 31), which enable the extraction of SVC data from other tracks by reference. The extractor can represent a concrete operation point of the SVC stream, identified by the top value of the DQT triple in the NAL units referenced by the extractor. In section III-B there is a more detailed discussion about this adaptation instrument.

Tiers are a more complex mechanism that permit to select operation points with more flexibility. A tier can represent a set of operation points using special boxes to group samples in an SVC video elementary track and to map groups of samples to scalable layers. The characteristics of the operation points (spatial resolution, bitrate, etc.) are stored in the tier definition, and can be used to take the adaptation decision.

SVC metadata tracks represent the more flexible and complex adaptation mechanism defined in the SVC file format. This type of tracks contain time parallel metadata associated with the SVC media data. Metadata is structured as statements that describe the scalability features of the video data, and can help in the implementation of complex extractors, specially when working with irregular SVC streams.

III. SVC TRACKS

A. SVC video tracks

In this section we describe the procedure to store SVC video elementary streams in SVC tracks. The file format specification permits to split the stream in several tracks. For instance, it could be interesting to store the base layer, AVC compatible, in one track, and the rest of enhancement layers in a second track. However, we have chosen to use a single track to store all the SVC stream, because extractor tracks (introduced below) can be employed to obtain similar results.

To keep back compliance with AVC file format, a track with a complete SVC stream must be described with the coding name 'avc1'. An AVC player can open this track and feed the decoder with all the SVC stream; the decoder must discard all the not understandable NAL units, in this case the SVC NAL units.

Regarding the sample description box *stsd*, the SVC file format makes use of the inheritance mechanism of the ISO Base Media file format to define a new sample description atom that contains, in separate sub-atoms, the AVC configuration (sequence and picture parameter sets) and the SVC configuration (sequence, subset sequence and picture parameter sets). An AVC player can retrieve the AVC configuration as usual, whereas an SVC player can use the SVC configuration to prepare the decoder. Additionally, this box stores a SEI NAL unit containing only a scalability information message. The scalability information includes the spatial, temporal and quality dimensions of each operation point as well as its bitrate requirements. This SEI message is fundamental to take decisions on operation point selection.

Samples in the SVC track correspond to SVC access units, that is, each sample contains information of a complete picture, including all the spatial and quality enhancement layers. NAL units of the access unit are stored in the order found in the original SVC stream. Usually, the access unit starts with a SEI NAL unit containing a sub-sequence information message, followed by a set of AVC VCL NAL units containing the base layer data (each one preceded by an SVC prefix NAL unit), and finally concludes with a set of SVC VCL NAL units representing the enhancement layers of the picture.

B. SVC extractor tracks

Extractor tracks permit the pre-filtering of SVC streams in order to reduce the server overload due to adaptation during streaming time. An extractor track is identified with the coding name 'svc1', and points to the original media tracks in its *tref* box with the 'scal' reference type. The samples in the extractor have a format similar to samples in AVC tracks, except for a special extractor NAL unit (type 31). Below we :

```
aligned(8) class Extractor () {
   NALUnitHeader();
   unsigned int(8) track_ref_index;
   signed   int(8) sample_offset;
   unsigned int((LSMO+1)*8) data_offset;
   unsigned int((LSMO+1)*8) data_length;
}
```

where NALUnitHeader() is an extended NAL unit header as specified in [14] and represented in Figure 1.

The track_ref_index in the extractor specifies the index of the 'scal'-type reference in the *tref* box of the extractor track, that points to the SVC media track from which data is extracted. The sample_offset gives a relative index of the sample in the SVC media track to be used as source in the extraction process. The relative positioning expressed in this field corresponds to the decoding time line of the extractor track. Usually it has value 0, indicating that the sample in the media track to be used has the same decoding time as the sample in the extractor track containing the extractor NAL unit. The extractor NAL unit is completed with the data_offset field, that indicates the offset of the first byte in the referenced sample to copy, and the data_length field, that contains the number of bytes to extract. The values in the SVC NAL unit header are obtained from the values in the original SVC media sample. The most important fields are did (dependency identifier), tid (temporal identifier) and qid (quality identifier). They contain the lowest values of the fields in the extracted NAL units.

The extractor track is created with a user selected interval of operation points, specified by a pair of DQT triples: DQT_{min} and DQT_{max}. Obviously, the DQT_{min} value

 ISBN: 978-1-61208-275-2

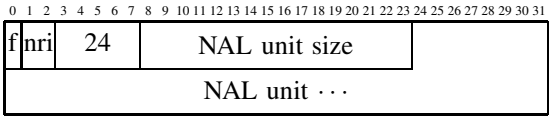

Figure 2. RTP payload field format

can specify the $\{0, 0, 0\}$ value, indicating that the extractor selects the SVC video from the base layer up the DQT_{max} operation point.

The sample description of the extractor track stores an SVC configuration box and a scalability info SEI box. This two boxes are filled with data extracted from the original SVC media track. More specifically, the SVC parameter sets included in the extractor sample description are those used by the NAL units pointed from the extractor track. Regarding the scalability info SEI box, the extractor contains a new scalability info message that describes only the layers in the interval of operation points specified by the user.

C. SVC hint tracks

The Network Abstraction Layer for AVC was defined having in mind the use of transport protocols to convey AVC data. In fact, the RTP payload format for AVC establishes that the first byte in the payload field of RTP packets mimics the first byte of a NAL unit. In the most simple configuration of the RTP packetization, each NAL unit can be transported in a single packet, so that the payload is exactly the complete NAL unit. This procedure is convenient when NAL units have an amount of bytes that is nearly the MTU parameter of the network (e.g., 1500 bytes in Internet, including RTP and UDP headers). Nevertheless, this choice is not always available, because the control of NAL unit size is done during the encoding process (usually by limiting the size of slices), but the encoding and hinting stages could be done by different agents.

If small size NAL units are transported individually in RTP packets, the protocol overload will be unacceptable. To cope with this situation, the AVC and SVC RTP payload formats define the use of a special type of RTP packets, called aggregation units. Two type of aggregation units were considered, Single-Time Aggregation Units (STAP) and Multi-Time Aggregation Units (MTAP). In this document we will focus on STAP in which all the NAL Units within a packet pertain to the same decoding instant.

Single Time Aggregation units are identified with the type field in the first byte of the RTP payload: its value is equal to 24. In this case, the RTP packet can contain several NAL units corresponding to the same access unit. Figure 2 shows the format of the payload field in these RTP packets.

The f field in the packet header is set if any of the aggregated NAL units has this field set. The nri field has the greatest value of the corresponding fields in the aggregated NAL units. Following this byte there is a sequence of NAL units, each one preceded by a 16 bit field that specifies its length.

It is also possible to have NAL units whose length exceeds the MTU parameter. In this case, the hinter is obliged to fragment the NAL unit into several consecutive RTP packets with identical timestamp. Each RTP packet carries a fragment of the original NAL unit. These RTP packets have the structure shown in Figure 3.

The Fragmentation NAL unit header has a type value of 28. The following 8 bits contain a special header with the fields:

- f: has the value of the same field in the fragmented NAL unit.
- nri: has the value of the same field in the fragmented NAL unit.
- s: start bit. Is set when the RTP packet carries the first segment of the fragmented unit.
- e: end bit. Is set when the RTP packet carries the last segment of the fragmented unit.
- r: reserved. Must be 0.
- type: contains the type of the original NAL unit.
- FU payload: contains a fragment of the NAL unit.

To finish this section, we discuss the inclusion of SDP presentation descriptions in SVC hint tracks, because it is a basic piece of information to carry out the adaptation. As mentioned previously, there are two levels where SDP data can be stored: the movie level, and the hint track level. When an RTSP client sends a DESCRIBE operation to an RTSP server, specifying an MP4 movie, the latter returns the complete SDP description of the file. The complete description joins the movie level SDP with the hint tracks level SDP (there could be more than one).

It has to be taken into account that the streaming scenario (RTSP) calls for a declarative use of session descriptions, that is, the SDP description sent from the streaming server as a response to the DESCRIBE operation represents session properties that can not be modified by a client counter proposal. In this context, the SDP description of a hint track contains three parameters that have a special importance during the adaptation process of the RTP stream generated from the track: *packetization-mode*, *sprop-parameter-sets* and *sprop-scalability-info*.

The packetization-mode parameter specifies the type of packets used in the hint track. If this parameter takes the value 0, RTP packets must carry single NAL units. When the value is 1, aggregation and fragmentation packets can be

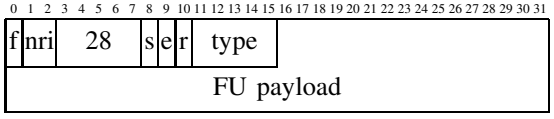

Figure 3. Fragmentation Unit payload field format

used, with the restriction of carrying information of a single access unit. When the value is 2, NAL units of multiple access units can be transported in the same RTP packet. As said above we will consider up to mode 1.

To include sequence, subset sequence and picture parameter sets in the SDP description, the hinter uses sprop-parameter-sets [9]. The value of this parameter is a comma separated list of these parameter sets represented in Base64 format. This list of parameter sets is collected from the sample description boxes of the extractor track or SVC media track.

Finally, the sprop-scalability-info is used to convey the scalability information SEI message in Base64 format. Again, the SEI message is retrieved from the sample description boxes of the referenced track.

IV. Sender driven adaptation

With the support of the SVC file format described in the previous sections, it is relatively straightforward the implementation of the server driven adaptation in the context of video streaming. In this scenario the server takes the adaptation decision based on static and dynamic context information. Static context information refers to the information relative to the whole session and stays immutable during it, such as client screen size. On the other hand, dynamic context is considered as the mutable information to be altered during the streaming time, such as the instantaneous available bandwidth.

It is important to consider that the MP4 file could have extractors for some (not all) of the possible intervals of operation points $[DQT_{min}, DQT_{max}]$. The server performs the selection in two steps: first, the scalability information message located in the single SVC track, which contains the complete representation of the video, is analysed to find the best operation point that complies with the static context. This operation point is called DQT_{target}; then, the server searches the best extractor (with greater DQT_{max}) that fulfils the conditions:

$$\begin{aligned} DQT_{min} &= \{0,0,0\} \\ DQT_{max} &<= DQT_{target} \end{aligned}$$

Once the server has selected a hint track for the streaming session, it prepares an SDP description that includes only the media level SDP sub-string corresponding to the selected hint track. This implies that the client is forced to ask for the SETUP of the unique hint track proposed by the server.

Dynamic fluctuations of the context (like network conditions) can oblige to perform a thinning of the RTP stream obtained from the selected hint track. To accommodate to the new conditions, the streaming server periodically renews the target operation point. If the operation point moves below the value of the hint track selected during the session setup, the RTP stream is forced to cross an SVC filter [15]. The filter de-packetizes the NAL units, removes those whose DQT identifiers aren't between the dependencies of the selected operation point, and re-packetizes the NAL units that pass the filter.

Dynamic adaptations can be performed at a MANE between the server and the client. The procedure to implement the dynamic adaptation is similar to the one described previously for the streaming server. It is important to note that the MANE has access to the scalability information that describes the input RTP stream, because this information is included in the SDP sprop-scalability-info parameter that is intercepted during the RTSP conversation between client and server.

V. Receiver driven adaptation

To implement the receiver driven adaptation, some additional information must be included in the SDP boxes of the MP4 file: the dependencies between hint tracks created from extractors. As mentioned in section III-B, extractors can be created using intervals of operation points $[DQT_{min}, DQT_{max}]$. Suppose an administrator creates n SVC extractors in an MP4 file with intervals defined by:

$$I(i) = \left[DQT^i_{min}, DQT^i_{max}\right] \quad (1)$$

such that:

$$\bigcup_{i=0}^{n} I(i) = [\{0,0,0\}, \{D_{max}, T_{max}, Q_{max}\}] \quad (2)$$

where D_{max}, T_{max} and Q_{max} represent the maximum scalability dimensions of the complete SVC stream. Then, the streaming client must be able to detect the decoding dependencies between extractors with $DQT_{min} \neq \{0,0,0\}$ in order to be able to setup enough RTP streams to convey a certain overall operation point.

The SDP extension defined in [16] is the choice of the RTP payload format for SVC [5] to describe the decoding dependencies between media elements defined in an SDP session. The signalling of dependencies proposed by the IETF is described with the following example:

```
v=0
o=john 2890844526 2890844526 IN IP4 10.10.0.1
s=SVC Receiver Driven
c=IN IP4 10.10.0.1
t=0 0
a=group:DDP L1 L2
m=video 20000 RTP/AVP 96
a=control:trackID=4
a=rtpmap:96 H264/90000
a=fmtp:96 profile-level-id=4d400a;
  packetization-mode=1;
  sprop-parameter-sets={sps0},{pps0};
  sprop-scalability-info={si0};
a=mid:L1
m=video 20002 RTP/AVP 97
a=control:trackID=5
a=rtpmap:97 H264-SVC/90000
a=fmtp:97 profile-level-id=53000c;
  packetization-mode=1;
  sprop-parameter-sets={sps1},{pps1};
  sprop-scalability-info={si1};
a=mid:L2
a=depend:97 lay L1:96
```

At the session level, the grouping type 'DDP' specifies the labels of the video tracks that have a decoding dependency. Labels are assigned to media identification attributes ('mid') at the media level. When a specific media can be decoded only when other media it depends on are available, the 'depend' attribute specifies this dependency (called layered dependency) listing the labels and payload type values of the required media. In the previous example, the first media section describes the base layer of an SVC video (AVC compliant). The second media section specifies an SVC enhancement that depends on the base layer.

The control attributes in the media sections specify the track identifiers that the client can use to setup independently each stream. Thus, a multi-interface scenario can be implemented using different client addresses (i.e., network interfaces) during the RTSP setup of each hint track. The receiver must inspect the complete SDP description of the file and infer the set of sessions that must be started to receive the best possible operation point, that is, the operation point nearer to the target operation point that can be obtained with the addition of one or more tracks. The scalability information obtained with the combination of the sprop-scalability-info parameters can be used in addition to the static context in order to calculate the target operation point. A receiver driven layered multicast scenario follows a similar procedure, but in this case the media level SDP contains the multicast address to which the receiver can subscribe in order to receive the corresponding SVC media.

VI. SCALNET SVC FILE FORMAT TOOL

During the SCALNET project [17], a file format tool implementing the features described above was developed. The *mp4creator* open source software was extended to support the creation of SVC media tracks, SVC extractor tracks and SVC hint tracks. In addition to this, modifications to Darwin Streaming Server and MPlayer were carried out to implement the sender and receiver driven adaptation scenarios, respectively. In the server side, these modifications included the implementation of an adaptation decision-taking engine to adapt the streaming sessions to the adequate operation point dynamically. An SVC filter was included in the Darwin Streaming Server to do fine-grained rate-shaping at the server side. Additionally, the filter was also employed in a MANE implemented as an RTSP proxy.

In comparison with other MP4 file creation tools, the SCALNET SVC file format tool includes the capability to create extractor tracks and dependency descriptions in the SDP metadata. Hint tracks permit to perform a coarse-grain off-line adaptation with a low overhead in the space occupied by the MP4 file. Without the availability of extractor tracks, the administrator of the streaming server has to create one media track for each offered operation point, and its corresponding hint track. To exemplify this, consider an SVC stream that occupies 69058031 bytes, with 1500 frames, two spatial layers with QCIF and CIF resolutions, five temporal layers, and three quality layers. The following list shows some of the values of each combination:

Layer	Resolution	Framerate	Bitrate	DTQ
0	176x144	1.5625	0.40	(0,0,0)
1	176x144	3.1250	8.30	(0,1,0)
2	176x144	6.2500	13.50	(0,2,0)
3	176x144	12.5000	17.40	(0,3,0)
4	176x144	25.0000	21.00	(0,4,0)
5	176x144	1.5625	61.10	(0,0,1)
6	176x144	3.1250	88.50	(0,1,1)
7	176x144	6.2500	103.00	(0,2,1)
8	176x144	12.5000	286.80	(0,3,1)
9	176x144	25.0000	551.20	(0,4,1)
10	176x144	1.5625	208.10	(0,0,2)
11	176x144	3.1250	318.70	(0,1,2)
12	176x144	6.2500	420.20	(0,2,2)
13	176x144	12.5000	1246.40	(0,3,2)
14	176x144	25.0000	2439.00	(0,4,2)
15	352x288	1.5625	317.20	(1,0,0)
16	352x288	3.1250	447.00	(1,1,0)
17	352x288	6.2500	558.10	(1,2,0)
18	352x288	12.5000	1570.40	(1,3,0)
19	352x288	25.0000	3090.00	(1,4,0)
...	...	...	...	...
29	352x288	25.0000	9206.00	(1,4,2)

Using the MP4Box tool with the following command:

```
MP4Box -add svcfile:svcmode=splitall mp4file
```

we create a media track per spatial and quality combination. This means six operation points, each one stored in a media track. Using the MP4Box `-hint` option we obtain the corresponding six hint tracks, one per media track. The result is a file with 73840255 bytes, representing a 6.2% overhead with respect to the original file.

Now we use the SCALNET SVC file format tool with one media track, and six extractor tracks and hint tracks, to provide the server with exactly the same offer of operation points. These are the commands used:

```
mp4creator -rate=25 svcfile mp4file
mp4creator -x=0,0,0-0,4,0 mp4file
mp4creator -x=0,0,0-0,4,1 mp4file
mp4creator -x=0,0,0-0,4,2 mp4file
mp4creator -x=0,0,0-1,4,0 mp4file
mp4creator -x=0,0,0-1,4,1 mp4file
mp4creator -x=0,0,0-1,4,2 mp4file
mp4creator -hint=2 mp4file
mp4creator -hint=3 mp4file
mp4creator -hint=4 mp4file
mp4creator -hint=5 mp4file
mp4creator -hint=6 mp4file
mp4creator -hint=7 mp4file
```

The first call creates a new mp4file starting with an SVC media track with the complete sequence at 25 frames per second. Each call to mp4creator with the `-x` option performs the creation of an extractor track with the range of layers specified. Take into account that the tool employs a DTQ triple instead of DQT. The subsequent calls with the

 ISBN: 978-1-61208-275-2

`-hint` option create a hint track for each extractor. The value indicated in this option specifies the track to hint. We obtain a file with 72953209 bytes, which represents a 5.6% overhead. The save in disk space depends on the characteristics of the SVC sequence, but such reduction can represent a big amount of space when the server provides an large collection of files.

In order to generate a file suitable for a receiver driven adaptation scenario, we can create extractors with :

```
mp4creator -x=0,0,0-0,4,0 mp4file
mp4creator -x=0,0,1-0,4,1 mp4file
mp4creator -x=0,0,2-0,4,2 mp4file
mp4creator -x=1,0,0-1,4,0 mp4file
mp4creator -x=1,0,1-1,4,1 mp4file
mp4creator -x=1,0,2-1,4,2 mp4file
# Hint tracks as in the previous example
mp4creator -R=6,8,9,10,11,12,13 mp4file
```

We have changed the extractor intervals and issued an extra command to create the DDP dependencies as explained above, where 6 is the number of hint tracks to be taken as source and 8, 9, 10, etc., are the hint tracks to be used to create the dependencies in the SDP description at the file level. The result is a file with 71135866 bytes, only a 3% overhead, which is lower than the sender driven adaptation approach as a consequence of the use of fewer pointers in the extractor tracks.

VII. Conclusions and Future Work

This paper provides an overview of how to use the SVC file format to implement two types of adaptation, server and receiver driven, in scalable video streaming services. A SVC file format tool is presented, incorporating new features with respect to previous ones, and some results demonstrate the reduction in the amount of disk space overhead obtained with the use of extractor tracks. The usability of the tool was tested in a complete streaming architecture.

The SVC file format tool can be improved to include other adaptation aids proposed in the standard, including tiers and metadata statements. The intention of partners involved in the development of this tool is to offer it as open source software in order to support the research and dissemination of the SVC standard.

Acknowledgements: This work was supported by the Spanish Ministry of Industry, Tourism and Commerce (MITYC) in the context of the Celtic SCALNET (CP5-022) and EU-ICT OPENLAB (FP7-287581) projects.

References

[1] D. Singer, editor. *ISO/IEC 14496-12:2005 Part 12: ISO Base Media File Format*. International Organization for Standardization, 2005.

[2] D. Singer, M. Zubair Visharam, Y. Wang, and T. Rathgen, editors. *ISO/IEC 14496-15:2004/Amd2: SVC File Format*. International Standardization Organization, 2007.

[3] H. Schwarz, D. Marpe, and T. Wiegand. Overview of the Scalable Video Coding Extension of the H.264/AVC Standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 17(9):1103–1107, September 2007.

[4] C. Perkins. *RTP. Audio and Video for the Internet*. Addison-Wesley, 2003.

[5] S. Wenger, Y. Wang, T. Schierl, and A. Eleftheriadis. RTP Payload Format for SVC Video. Technical report, Internet Engineering Task Force, May 2011. Standard, RFC 6190.

[6] *ISO/IEC 14496-14:2003 Part 14: MP4 File Format*. International Standardization Organization, 2003.

[7] *ISO/IEC 14496-15:2004 Part 15: Advanced Video Coding (AVC) file format*. International Standardization Organization, 2004.

[8] P. Amon, T. Rathgen, and D. Signer. File Format for Scalable Video Coding. *IEEE Transactions on Circuits and Systems for Video Technology*, 17(9):1174–1185, September 2007.

[9] S. Wenger, M.M. Hannuksela, T. Stockhammer, M. Westerlund, and D. Singer. RTP Payload Format for H.264 Video. Technical report, Internet Engineering Task Force, 2010. draft-ietf-avt-rtp-rfc3984bis.

[10] S. McCanne, V. Jacobson, and M. Vetterli. Receiver-driven layered multicast. In ACM, editor, *SIGCOMM '96: Conference proceedings on Applications, technologies, architectures, and protocols for computer communications*, pages 117–130, 1996.

[11] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications. Technical report, Internet Engineering Task Force, July 2003. Standard, RFC 3550.

[12] H. Schulzrinne, A. Rao, and R. Lanphier. Real Time Streaming Protocol (RTSP). Technical report, Internet Engineering Task Force, April 1998. Proposed Standard, RFC 2326.

[13] M. Handley, V. Jacobson, and C. Perkins. SDP: Session Description Protocol. Technical report, Internet Engineering Task Force, July 2006. Proposed Standard, RFC 4566.

[14] T. Wiegand, G. Sullivan, H. Schwarz, and M. Wien, editors. *ISO/IEC 14496-10:2005/Amd3: Scalable Video Coding*. International Standardization Organization, 2007.

[15] M. Ransburg, E. Martínez Graciá, T. Sutinen, J. Ortiz, M. Sablatschan, and H. Hellwagner. Scalable video coding impact on networks. *Workshop SVCVision - Mobimedia*, 2010.

[16] T. Schierl and S. Wenger. Signaling Media Decoding Dependency in the Session Description Protocol. Technical report, IETF, 2009.

[17] Michael Ransburg, Eduardo Martínez, Tiia Sutinen, Jordi Ortíz, Michael Sablatschan, and Hermann Hellwagner. Scalable video coding impact on networks. In *Mobile Multimedia Communications*, volume 77 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 571–581. Springer Berlin Heidelberg, 2012.

 ISBN: 978-1-61208-275-2