An Architecture for Distributing Scalable Content over Peer-to-Peer Networks

Nicola Capovilla[1], Michael Eberhard[2], Silvano Mignanti[3], Riccardo Petrocco[4], and Janne Vehkaperä[5]

[1]STMicroelectronics, Milan, Italy, nicola.capovilla@st.com
[2]Klagenfurt University, Klagenfurt, Austria, michael.eberhard@itec.uni-klu.ac.at
[3]University of Rome 'Sapienza', Rome, Italy, silvano.mignanti@dis.uniroma1.it
[4]Technische Universiteit Delft, Delft, Netherlands, r.petrocco@gmail.com
[5]VTT Technical Research Centre, Oulu, Finland, janne.vehkapera@vtt.fi

***Abstract*—Peer-to-Peer systems are nowadays a very popular solution for multimedia distribution, as they provide significant cost benefits compared with traditional server-client distribution. Additionally, the distribution of scalable content enables the consumption of the content in a quality suited for the available bandwidth and the capabilities of the end-user devices. Thus, the distribution of scalable content over P2P networks is a very actual research topic. This paper presents an architecture for the distribution of scalable content in a fully distributed Peer-to-Peer network. The architectural description includes how the scalable layers of the content are mapped to the pieces distributed in the Peer-to-Peer system and detailed descriptions of the producer- and consumer-site architecture of the system. The presented system is to our knowledge the first open-source Peer-to-Peer network with full Scalable Video Coding support.**

Keywords-Error Concealment; Packetizing; Peer-to-Peer; Scalable Video Coding; Scalability Metadata

I. INTRODUCTION

The streaming of content over Peer-to-Peer (P2P) networks becomes more important as the popularity of Internet multimedia services is increasing and the corresponding server costs are rising as well. One of the major challenges of distributing multimedia content is that different users often require the content in different quality. This is due to the differences in the users' network connections and the capabilities of their terminals. A solution for distributing layered content in different qualities within one bitstream is provided by the Scalable Video Coding (SVC) extensions of the Advanced Video Coding (AVC) standard [1].

In this paper we are going to describe an architecture for the distribution of scalable content in a fully distributed P2P network. The P2P system targeted for the integration is the NextShare system, which is developed within the P2P-Next project [2]. P2P-Next is a research project partially founded by the European Commission in the context of the Framework Program 7, within the ICT (Information and Communication Technology) theme. The main goal of P2P-Next is the development of an open-source next generation P2P content delivery platform, the NextShare system.

The NextShare system has been developed based on the Bittorrent protocol [3] and thus provides an implementation of a fully distributed P2P system. To support Video on Demand (VoD), live streaming and the distribution of scalable content in the NextShare system, a number of modifications to the original Bittorrent protocol have been performed [4].

One of the main reasons for implementing SVC support for the NextShare system is that there is to our knowledge today no open-source P2P system supporting SVC available that can be downloaded and tested by interested users. The advantages of distributing scalable content compared to simulcast approaches have been evaluated in a number of surveys (see, e.g., [5]).

Although the distribution of SVC content over P2P systems has been addressed in the literature before, other P2P/SVC solutions that provide a good theoretical understanding of the problem, like LayerP2P [6], do not utilize real SVC codecs for their prototype implementation but rely on the usage of H.264/AVC-compatible codecs that can only be used to test one of SVC's scalability dimensions, the temporal scalability. Thus, one of the goals of the NextShare implementation was to design, implement, and distribute an open-source system with full SVC support.

The remainder of this paper is organized as follows: In Section II, the approach for the integration of the scalable content into the NextShare system is described. In the following two sections, the producer- and consumer-site of this architecture are described in detail. Finally, future work is addressed in Section V and Section VI concludes the paper.

II. NEXTSHARE INTEGRATION

To fully integrate scalable content into the NextShare system, a number of problems had to be addressed. Two main problems, the selection of suitable scalability layers and the mapping of the layers to Bittorrent pieces, are described in detail within this section. While the selection of the scalability layers tries to consider all popular qualities and to support a number of different network connections, the mapping to the scalability layers to the Bittorrent pieces

tries to ensure that the best trade-off between flexibility for possible quality switches and overhead in terms of piece management is found.

It should be noted that even though we are using SVC within our NextShare system, all design decisions have been made with the intention to make the architecture codec-agnostic. Thus, if another scalable video codec is utilized within the NextShare system, only the coding and packaging tools need to be replaced, while the integration into the NextShare core will remain suitable for every other layered codec.

A. *Scalability Layers*

The first step for the integration of scalable content into the NextShare system was the selection of the desired scalability layers. The selected layers are illustrated in Table 1.

TABLE 1: SCALABILITY LAYERS

Bitrate	Resolution	Quality	frames/sec
512 Kbps	320x240	low	25
1024 Kbps	320x240	high	25
1536 Kbps	640x480	low	25
3072 Kbps	640x480	high	25

As illustrated in Table 1, four scalable layers were selected for the integration. The main reasons for selecting this layer structure were to maintain a good coding efficiency and to provide all popular qualities. The possibility to add further layers to support HD content is also fully supported by our framework, but has been omitted for the current version due to constraints in the upload bandwidth of our system's users. From the coding-efficiency point of view, the difference between the layers in terms of bit rate should be not too low, as the coding efficiency decreases drastically in such cases [1], while the selected bit rates represent the most popular qualities that are provided nowadays by multimedia portals. Furthermore, it should be noted that the audio bitstream is provided together with the video bitstream of the base layer. Thus, the 512 Kbps for the base layer include the bit rate for the 128 Kbps audio bitstream.

B. *Mapping to Bittorrent Pieces*

The second step of the integration process is the mapping of the scalability layers to Bittorrent pieces. Firstly, the unit shall represent a synchronization point for dynamic switches between different quality layers. To achieve this goal each unit starts with an Instantaneous Decoding Refresh (IDR) reference frame. Secondly, it should be noted that we do not perform a direct mapping to pieces but to a unit. This unit represents a fixed number of frames for a specific layer and can be mapped to a fixed number of pieces. The reason for this approach is that the piece size might be changed in the P2P system for various reasons, and by basing the mapping on units rather than on pieces only the unit/piece-mapping needs to be updated when the piece size is modified.

The mapping to the units has been performed based on several criteria. First, the units need to be selected large enough to allow for a good coding efficiency. As it should be possible to decode each unit independently (when all lower layer units for the same time stamp are also available) the number of frames within one unit should be high enough to allow for good coding efficiency. Additionally, the number of frames within one unit should be low enough to provide the flexibility to conveniently switch between qualities when the network conditions change.

Based on these considerations, a mapping of 64 frames, which represents 2.56 seconds of content at a frame rate of 25 frames/sec, has been selected. Such a unit is subsequently mapped to three pieces. However, as noted previously, the piece mapping can always be changed based on the requirements from the P2P system. Additionally, the number of frames per unit might also be changed, e.g., to 128 frames, if a slightly better coding efficiency but lower flexibility is desired. The mapping is illustrated in Table 2.

TABLE 2: UNIT MAPPING

Layer	Kb/time slot	KB/time slot	pieces/time slot
BL	512 Kbps * 2.56 ≈ 1.310	/ 8 ≈ 164 KByte	3 pieces @ 55 KByte/time slot
EL1	1024 Kbps * 2.56 ≈ 2.621	/ 8 ≈ 328 KByte	6 pieces @ 55 KByte/time slot (3 pieces in previous layers, 3 new pieces)
EL2	1536 Kbps * 2.56 ≈ 3.932	/ 8 ≈ 492 KByte	9 pieces @ 55 KByte/time slot (6 pieces in previous layers, 3 new pieces)
EL3	3072 Kbps * 2.56 ≈ 7.864	/ 8 ≈ 983 KByte	18 pieces @ 55 KByte/time slot (9 pieces in previous layers, 9 new pieces)

The mapping to the 55 KByte pieces results in a small overhead of available bits per piece. However, this overhead is utilized to compensate the small drifts of the constant bit rate (CBR) algorithm utilized during the SVC encoding process.

III. PRODUCER-SITE ARCHITECTURE

The producer-site architecture describes all steps from encoding the SVC bitstream to the ingestion into the core of the P2P system. The topics addressed in this section include the encoding process, the splitting of the bitstream, creating metadata based on the bitstream's supplemental enhancement information (SEI), packetizing the bitstream, and ingesting the bitstream into the core of the P2P system. An illustration of this architecture is provided in Figure 1, more details on each of the processing steps are provided in the following sections.

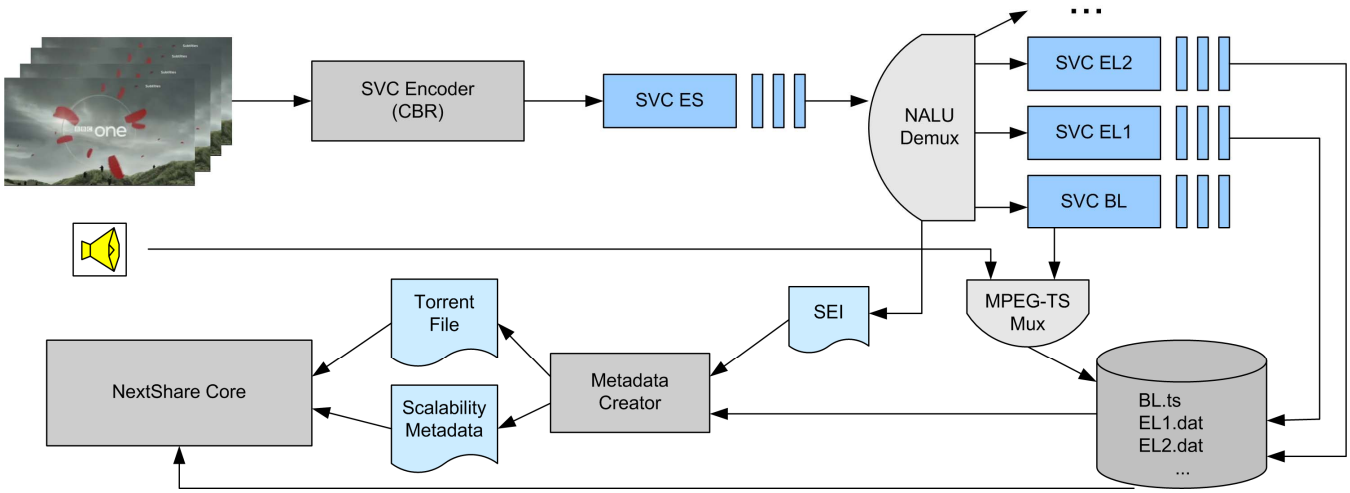

Figure 1. Producer-Site Architecture.

A. *Bitstream Preparation*

As the first step of the bitstream preparation process, the raw video (i.e., the YUV video frames) is encoded by an optimized (mainly, the motion estimation algorithm was optimized) JSVM 9.15 [7] encoder, which uses a CBR algorithm to ensure that the pieces created from the video content have a constant size. The audio data can be either provided already encoded, e.g., as an MP3 or AAC audio file, or transcoded to the desired audio format if a raw PCM audio file is provided.

The encoded SVC bitstream is subsequently split into the H.264/AVC-compatible base layer (BL) and the enhancement layers (EL) by the Network Abstraction Layer Unit (NALU) demuxer. The demuxer analyzes the NALU headers and splits the access units into separate bitstreams for each layer. Additionally, the SEI information at the beginning of the bitstream (i.e., the scalability_info message) and the Sequence Parameter Set (SPS) and Picture Parameter Set (PPS) are provided to the metadata creator (see Section C).

B. *Bitstream Packetizing*

In the bitstream packetizing step, the base layer of the SVC bitstream is muxed with the audio into an MPEG-2 Transport Stream (MPEG-TS). The main reason for this step is that the base layer should be provided in a backwards-compatible way, so that also end user terminals that only support H.264/AVC can successfully process the base layer. MPEG-TS is a standard able to encapsulate audio and video Packetized Elementary Streams (PESs) and other data and is defined in [8].

An MPEG-TS file contains a continuous stream of TS packets. The size of every TS packet is 188 bytes, to achieve the desired mapping of TS packets to one unit its size must be an exact multiple of 188 bytes. Considering the video bitstream provided by the SVC encoder could have a lower bit rate than the target bit rate (due to small drifts of the CBR algorithm), padding would be necessary during the packetizing process to ensure that the right amount of frames can be mapped to one unit. The MPEG-TS standard provides null packets for this purpose.

C. *Scalability Metadata Support*

Although the pieces of the video stream are transmitted over the network in a layered way, the de-packetizer at the consumer-site needs to know the properties of the layers for the decoding process and the decoder needs access to the parameters from the beginning of the bitstream (the SPS and PPs elements). Thus, the properties of the layers, which are usually provided by the Supplemental Enhancement Information (SEI) at the beginning of the bitstream, and the parameter sets need to be forwarded to the consumer-site.

To store these metadata and transmit them to the de-packetizer when needed, the SEI message and the parameters are forwarded from the NALU demuxer to the metadata creator. The metadata creator subsequently parses the SEI data and stores the properties of the layers in an XML metadata document. Additionally, the SPS and PPS elements are encoded in base64 and stored in the metadata document as well.

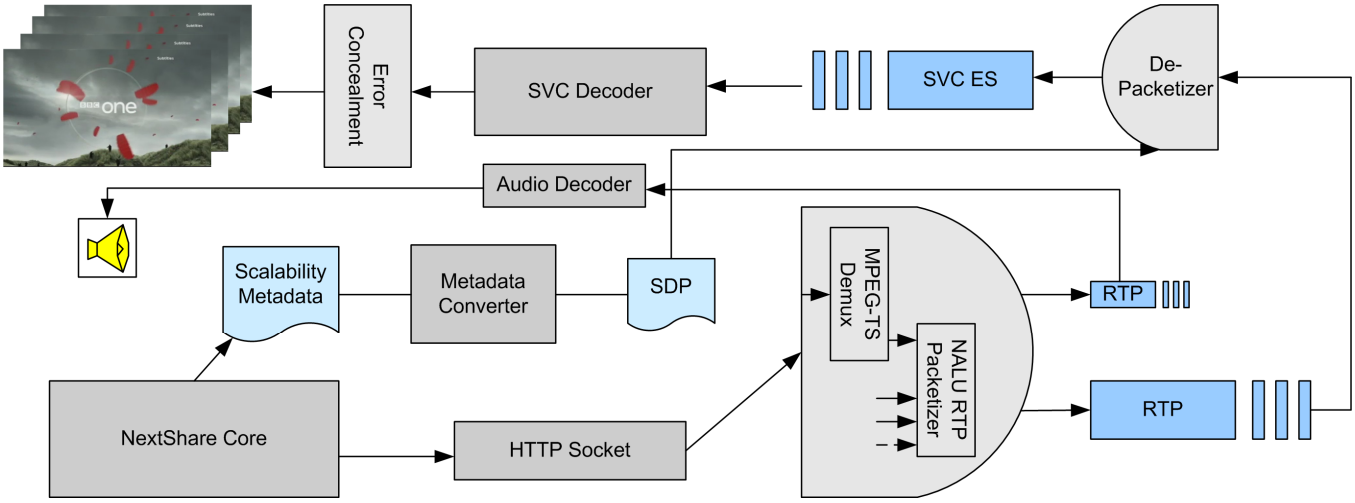

Figure 2. Consumer-Site Architecture.

D. *Ingest into the Core*

The NextShare core represents the P2P engine responsible for creating and injecting the content into the network. The main metadata file required for the ingestion of the content into the P2P system is the torrent file. The torrent file provides the information required for the download of the previously encoded base and enhancement layers, as well as metadata related to the content including the previously created scalability metadata. The created torrent file is compatible with the Bittorrent protocol [3] and therefore available for every peer running a compatible client, increasing the reliability of the torrent swarm and the scalability of the distribution costs (as the torrent file can not only be used by NextShare-compatible clients, but by all Bittorrent-compatible clients).

The fact, that the H.264/AVC-compatible base layer and the audio stream are provided commonly packetized into an MPEG-TS, enables also not SVC-enabled clients to join the swarm by being able to consume the stream in base layer quality. Therefore, every peer has an incentive to download at least the base layer, which increases its availability in the swarm.

After the creation of the torrent file, the content is injected into the NextShare core (i.e., the torrent file is distributed to other peers and the files containing the base- and enhancement layers are seeded). During the ingestion process, the base and enhancement layer files are split into pieces as illustrated in Table 2. During the mapping process some problems have to be taken into consideration. Firstly, the integrated CBR algorithm does not provide an exactly constant bit rate, but allows for minor drifts. Thus, the piece mapping is always performed with a small overhead. Additionally, the 188 byte size utilized for the MPEG-TS packets cannot be mapped to a power of two, while the pieces ingested into the NextShare core should have a multiple of two as their size. Thus, the possibility of choosing a piece size that differs from the standard power of two has been successfully investigated.

IV. Consumer-Site Architecture

The consumer-site architecture describes all steps from receiving the bit stream through the P2P system to decoding the bit stream for displaying it in the media player. The steps described include the local streaming of the received content, the de-packetizing of the content, the signaling of the layer properties using suitable metadata, and the merging and decoding of the received layers. An overview of this architecture is provided in Figure 2 and described in detail in the subsequent sections.

A. *Provision of the Received Content*

When the content is accessed by the user the layers from the NextShare swarm are downloaded in an intelligent way in order to maximize the Quality of Experience (QoE) for the current available download bandwidth. A great advantage of changing quality by displaying more or less enhancement layers regards the fall-back scenario: in a P2P system peers are considered to have an unreliable and selfish nature, leaving the swarm and decreasing the total available bandwidth as soon as they have received the desired content. In such a case the fall-back scenario will occur, where the user will experience a slow decrease of the QoE, as the download engine avoids downloading higher layers for upcoming time slots, if enough pieces of the previous layers are not already available.

Retrieving the content from the network is based on a modified approach of the Give-to-Get (G2G) algorithm [4]. The G2G algorithm selects the high-priority pieces based on their deadline, i.e., the piece with the nearest deadline in the high-priority set is downloaded first to ensure a continuous playback of the content. For the layered application of the G2G algorithm, the priority sets are applied to all the active available layers in a proportional way.

As discussed in the previous section, by providing backwards compatibility with other clients, the availability of the base layer at all peers in the same swarm can be assumed. Therefore, if a download bandwidth of at least the base

layer's bit rate is provided by the peer's connection, it can also be assumed that the playback will never stall (if the neighbor peer seeds the content or is ahead in its playback position). Note that once a peer finishes watching the content, the download engine will start retrieving the remaining pieces of all the layers for two major reasons: Firstly, to increase the layer's availability in the swarm, and secondly, to enable watching the content at the highest quality again once all the layers have been downloaded. This default behavior could be disabled by the user if, e.g., no repeated watching of the content is desired.

After enough content has been downloaded to guarantee a continuous playback of at least the base layer, the download engine of the NextShare core will initialize the demuxer module. The multimedia data is forwarded to the demuxer utilizing an HTTP socket, which was selected to ensure interoperability between the NextShare core and third-party de-packetizing/decoding solutions. A persistent connection will be established between the demuxer and the Next Share core, allowing the demuxer module to sequentially ask for the available content for the following time slot, depending on the requests it receives from the decoder module. It is important to notice that the available pieces of the following time slot will be sent to the demuxer as late as possible, allowing the NextShare core to manage the major buffer, to increase the quality until the last moment and to try to increase the quality if the user pauses the playback.

B. Bitstream Demuxing and Packetizing

As described before, the demuxer works online: it receives from the swarm at least the MPEG-TS stream of the base layer, which is processed by the MPEG-TS demuxer. The demuxer splits the MPEG-TS into the audio content and the H.264/AVC-compatible base layer.

The elementary audio stream is directly encapsulated into a Real-Time Transport Protocol (RTP) packet stream (e.g., according to [9] for an MP3 audio stream) and is sent to the successive module. The elementary SVC base layer video stream is forwarded to the NALU RTP packetizer, which adds the base layer and the received enhancements layers into an RTP packet stream, reordering the packets and forwarding the RTP stream containing all layers to the next module.

Please note that all the modules represented in Figure 2 are typically running on the same host, i.e., the peer that receives the content. However, the main reason for selecting the RTP protocol to convey the audio and video data was to provide flexibility and to enable a possible integration of the P2P network with a more traditional server-based network. In such a server-based network the NextShare consumer-peer could act additionally as a server, receiving the content from the P2P network and redistributing the scalable content within an RTP streaming network.

C. Scalability Metadata Support

To decode the layers received from the NextShare system, the de-packetizer needs to be aware of the scalability properties of these layers. To provide a generic signaling mechanism for these properties, which could also be used by third-party solutions, we have decided to provide this information as a Session Description Protocol (SDP) document to the de-packetizer. The SDP document is formatted according to [10], which provides the capabilities to signal the properties and dependencies of the scalable layers.

As the metadata is provided by the NextShare core in a NextShare-conforming XML format, the scalability metadata document is firstly converted to the targeted SDP format. Subsequently, the SDP message containing the layer properties and the parameters sets is forwarded to the de-packetizer.

D. Bitstream Consumption

After the demuxing and RTP packetizing of the audio and video content, the RTP streams are forwarded. While the audio stream is forwarded to a standard de-packetizer and decoder, the SVC RTP stream is processed by our customized tools.

Firstly, our SVC de-packetizer parses the RTP packets and provides the payload and the time stamps to the SVC decoder. To perform this extraction process, the de-packetizer needs to have access to the properties of the SVC layers. These properties are provided by the SDP description. Additionally, the de-packetizer forwards the parameter sets from the SDP to the decoder, which are required for the decoding process. Finally, our highly optimized version of the JSVM 9.15 reference decoder performs the real-time decoding of the SVC content.

E. Error Handling

The selected structure of layers presented in Table 1 supports two different resolutions for the video. In some cases the layers for high resolution video cannot be received within the defined time slot even though the consumer prefers higher resolution video. As an example, this could happen while streaming on a heavily congested access point. Missing high resolution layer(s) will eventually lead to either decoding failure or to varying video resolution (from high to low), which can be a very annoying phenomenon for human eye.

To cope with the missing spatial enhancement layers, we have integrated an up-scaling functionality into the decoder based on normative integer-based 4-tap filters. The target resolution is defined in the SPS NAL unit and is compared with the resolution received when decoding a new time-slot. If the resolutions do not match with each other, frames of the new time-slot are up-scaled from the lower resolution to the target resolution to maintain the preferred resolution.

V. FUTURE WORK

Although our architecture already provides a full solution for the streaming of scalable content over P2P networks, there are still open possible modifications that could enhance the viewing experience for the user.

Firstly, the current approach only utilizes the layered scalability provided by the SVC standard, i.e., the scalability

in terms of resolution and fidelity. As a further step an additional support of temporal scalability would be desirable. Such a support could be realized by reordering the frames according to the hierarchical prediction structure and by storing the frames for different temporal levels in different pieces.

Another desirable further step would be the support of medium-grain scalability. However, such a support would require some changes to our architecture. As the piece picking algorithm currently works on piece level, where each piece contains a number of NALUs, and the medium-grain scalability allows to change the quality on NALU level, a new algorithm has to be designed to allow the partial download of pieces (i.e., to work on chunk level, with the need to investigate compatibility with existing clients).

VI. CONCLUSION

In this paper an architecture for the provision of scalable content in a fully distributed P2P system was presented. In Section II, the selection of the scalable layers was explained and the mapping of the layers to units and subsequently Bittorrent pieces was illustrated. The main reasons for the choice of 64 frames per unit were to achieve a good coding efficiency while still maintaining the flexibility to quickly switch between layers if the network conditions change.

In Sections III and IV the producer- and consumer-site architecture were presented. The producer-site architecture includes the encoding of the SVC bitstream and splitting of the bitstream into layers, the packetizing of the base layer and the audio stream into an MPEG-TS, the creation of the scalability metadata based on the SEI information at the beginning of the bitstream, and the ingest of the content into the core of the P2P system. On the consumer-site, the modules for retrieving and consuming the content were presented. The retrieved content is provided to the demuxer utilizing an HTTP socket and the demuxed audio and video streams are forwarded to the media consumption solution using RTP. The de-packetizing and decoding of the SVC content is subsequently performed by our customized SVC tools utilizing the scalability information provided by a suitable SDP document.

The presented system is to our knowledge the first open-source P2P system with full SVC support. Additionally, it is fully compatible with third-party media consumption solutions due to utilizing the HTTP, RTP, and SDP protocols for providing the content from the P2P system to the media consumption modules. Thus, the system is easy to use and customize for interested users.

ACKNOWLEDGMENT

This work is supported in part by the European Commission in the context of the P2P-Next project (FP7-ICT-216217) [2].

REFERENCES

[1] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the Scalable Video Coding Extension of the H.264/AVC Standard", IEEE Trans. on CSVT, vol. 17, no. 9, September 2007, pp. 1103-1120.

[2] The P2P-Next Project, FP7-ICT-216217, http://www.p2p-next.org, accessed on 23/03/2010.

[3] B. Cohen, "Bittorrent Protocol 1.0", http://www.bittorrent.org/beps/bep_0003.html, accessed on 23/03/2010.

[4] J.J.D. Mol, J.A. Pouwelse, M. Meulpolder, D.H.J. Epema, and H.J. Sips, "Give-to-Get: Free-riding-resilient Video-on-Demand in P2P Systems", in Multimedia Computing and Networking, San Jose, USA, 2008, SPIE Vol. 6818.

[5] A. Sentinelli et al., "A Survey on P2P Overlay Streaming Clients", in "Towards the Future Internet - A European Research Perspective", IOS Press, 2009, pp. 273-282.

[6] Z. Liu, Y. Shen, K. W. Ross, S. S. Panwar, and Y. Wang, "LayerP2P: Using Layered Video Chunks in P2P Live Streaming", IEEE Trans. on Multimedia, vol. 11, no. 7, August 2009, pp. 1340-1352.

[7] JSVM 9.15 Software Manual, 2008.

[8] ISO/IEC 13818-1, Information technology - Generic coding of moving pictures and associated audio information: Systems, 2000.

[9] R. Finlayson, "A more loss-tolerant RTP payload format for MP3 audio", RFC 3119, 2001.

[10] T. Schierl and S. Wenger, "Signaling Media Decoding Dependency in the Session Description Protocol (SDP)", RFC 5583, 2009.