



EvalSVC - An evaluation platform for Scalable Video Coding transmission

Tien Anh Le, Hang Nguyen, Hongguang Zhang

Abstract—Scalable Video Coding is the latest extension of the famous Advance Video Coding standard. The main advantage of SVC is that it can provide scalability for visual services which are serving customers with heterogeneous network conditions and terminals' capabilities. Nevertheless, the multimedia service research community and industry have not been able to fully utilize the entire potential of this video coding standard extension. One important reason is because of the lack of an evaluation tool-set and platform widely available for usage in the designing, evaluating as well as deploying processes of SVC-based visual services. EvalSVC aims to fill this gap and fosters SVC-based applications and research in multimedia services. It is capable of evaluating the enhanced features (such as spatial, temporal, SNR, and combined scalability) of SVC bit-streams transmitting over real or simulated networks. This tool-set is publicly available.

Index Terms—Scalable video coding; video evaluation platform; QoS Metrics and Measurement; simulation platform;

I. INTRODUCTION

H.264/MPEG-4 Advance Video Coding (AVC)[1] is a very famous video compression standard. Its compression ratio has enabled many video communication services (such as video conferencing, video surveillance or video-phony...). However, a fatal limitation of this standard is that it is not scalable enough for many services. Once a source video stream has been encoded with AVC, that encoded bit-stream will remain the same throughout the communication process. Encoding parameters of the encoded bit-stream (such as bit-rate, frame-rate, screen size, SNR...) will be determined at the beginning of the communication session by senders and receivers (mostly by receivers). However, those senders and receivers may have different screen sizes, different computational capabilities, network conditions (such as bandwidth, delay, jitter...) may change during the communication session. In those cases, in order for the AVC encoded bit-stream to be consumed adaptively at each and every receiver, there must exist middle-boxes in the communication network to convert the incoming AVC encoded bit-streams into various output bit-streams which are suitable for each receiver. This causes a huge delay in the communication session and single points of failure in the communication network. Otherwise the bit-stream will be stuck at bottle-necks and the entire video communication session will be broken. All of these problems make AVC not scalable enough for many video communication services.

People are now working and entertaining in a "3-screen"

Authors are with the Department of Wireless Networks and Multimedia Services, Telecom Sud Paris, France, 91000. Phone: +33 (0)1 60 76 66 63, Fax: +33 (0)1 60 76 45 78, E-mail: tien_anh.Le@it-sudparis.eu. This work was supported in part by CAM4Home, an European project.

world. These screens are different in their computational capacities, screen resolutions, and communication bandwidths. A much better solution than AVC is to use Scalable Video Coding (SVC). SVC has been standardized as an extension of the AVC standard since 2007[2]. The main idea of this extension is to apply multi-layer coding into the AVC codec. This is not a totally new idea since people had attempted to implement this idea from prior international video coding standards such as H.262/MPEG.2 Video, H.263 and MPEG.4 Visual[2]. However, the most challenging problem is that, the scalability used to come with a huge increase in computational complexity. SVC has succeeded in providing scalability with an affordable computational cost. SVC encodes an input video stream into a multi-layer output bit-stream comprising of a base layer and several enhancement layers. Within those layers, the base-layer is encoded with a basic quality to guarantee that it can be consumed by the weakest receiver of the entire communication group. This base-layer is usually protected while being transmitted over the network by QoS assured transmission methods or Forward Error Correction (FEC) algorithms. For the purpose of backward compatibility, the base-layer must be recognized by all conventional H.264 decoders. Enhancement layers, when received at the receivers together with the base-layer, can enhance the overall-quality of the bit-stream. Especially, when all enhancement layers are received in-order at the receiver together with the base layer, the bit-stream will achieve its original encoded quality. However, when real conditions (such as bandwidths, delays, or displaying screen sizes) do not allow, upper layers can be discarded along the transmission link or at any middle box (relaying entities) for the bit-stream to be fit-in with those conditions without corrupting the video communication session.

Video services using SVC have been launched since the standardization of the SVC codec. In order to evaluate those services, designers and researchers are really in-need of a video transmission evaluation tool which is specially tailored for the evaluation of SVC encoded video transmissions over a real or simulated network. So far, the research community depends on Evalvid[3] for measuring the objective QoS-related parameters of the under-layer networks (such as loss-rate, delays, jitter...), as well as evaluating both the subjective (using Mean Opinion Score - MOS) and objective (Peak Signal to Noise Ratio - PSNR) video quality metrics. Evalvid has supported only up to the H.264 video codec. In[4], the connecting interface of Evalvid was extended to replace its simple error simulation model by a more general network simulator like NS-2 so that researchers and designers can plug their own network

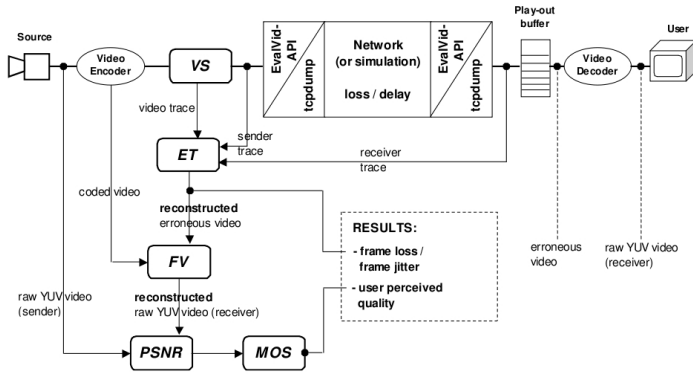


Fig. 1. Schematic illustration of the evaluation framework provided by EvalVid [3].

architectures in to evaluate. However, none of the above has taken SVC and its metrics into the evaluation.

In section 2, we will take a look at the existing video evaluation systems such as Evalvid and its extension (My-Evalvid). Section 3 will introduce scalable video coding emphasizing on the different characteristics of SVC toward AVC and some challenging problems to be overcome in order to integrate SVC into the existing video evaluating platforms. In section 4, we will evaluate performances of a SVC video transmission systems using realistic simulation network conditions with the help of our EvalSVC evaluation platform. Section 5 will conclude and open some possible future works based on this framework.

II. EVALVID AND ITS EXTENSION

Prior to Evalvid, other publicly available video evaluation tools[5][6] were struggling with synchronized frames between the sender and the receiver sides meaning that they could not evaluate the video quality in case of frame drops or frame decoding errors. Evalvid, with a modular structure design, overcame this major problem.

The main components of the Evalvid video evaluation framework (Fig. 1) are as follows:

Source: This can be a raw source video file or raw video stream captured from a working camera. Normally, an YUV file format is used as it is the common video format supported by almost all video capturing devices.

Video encoder and video decoder: Evalvid only supports a single layer video coding (mostly AVC).

Video sender (VS): After being compressed by a video coder, the video bit-stream will be fed to the video sender block of Evalvid. The main job of VS is to segment large video frames (or more precisely in AVC, Network Abstraction Layer - NAL frames) into a number of smaller UDP packets before feeding them to a real or simulated network. VS also logs video frame number, frame type, frame size, number of segmented UDP packet, and timestamps down to a video trace file. With the help of a TCP dumping tool, the video bit-stream sent by VS also generates a sender trace file. The video trace file and the sender trace file will

then be used in the video evaluation step. In order for VS to send a video bit-stream to the network, it needs hinting information of that bit-stream. The hinting information can be obtained by using the MP4Box tool of the GPAC framework[7]. That hint track also creates RTP (Real Time Protocol) packetization and payload format information based on the IETF's recommendations (RFCs) and/or standards of the video encoder (for the H.264 encoder, MP4Box uses RFC 3984[8]). MP4Box will also integrate that hint track and the video track into one common ISO compatible video file.

Evaluate Trace (ET): The evaluation process takes place at the sender's side. By using the video trace file, the sender trace file collected from the sender's side and a receiver trace file collected at the receiver's side, ET has the information about the timestamps, the packet ID, and the packet payload size. Based on this information, ET calculates the frame/packet loss and the frame/packet jitter for each frame type (I, B, P). Since a frame is comprised of several packets, it is considered lost when the packet loss rate over-floats a limit which a particular decoder can afford. Delay and jitter are also calculated. A very important function of ET is that it tries to reconstruct an output bit-stream at the receiver from trace files and the original encoded video bit-stream at the sender's side.

Video evaluation (PSNR and MOS blocks): Mainly, there are two approaches to measure digital video quality. They are subjective and objective quality metrics. For subjective measurement, Evalvid uses Mean Opinion Score (MOS)[9], which scales the human quality impression on the video from bad (0) to excellent (5). For objective measurement, Evalvid uses the Peak Signal to Noise Ratio (PSNR) frame by frame. In YUV video, since the human's eyes are more sensible with the luminance component of the video than with color components, Evalvid calculates PSNR of the luminance component Y of source image S and destination image D.

TABLE I
CONVERSION BETWEEN PSNR AND MOS[4].

PSNR [dB]	MOS
> 37	5 (Excellent)
31 - 37	4 (Good)
25 - 31	3 (Fair)
20 - 25	2 (Poor)
< 25	1 (Bad)

Fix Video: The MPEG standard defines three types of frames: I, B and P. P frames can only be completely decoded if the previous I or P frame is available. B frames also can only be decoded if the previous and successive I or P frame is available. That's why MPEG reorders the frames before transmission, so that any frame received can be decoded immediately. Because of this reordering issue, a coded frame does not corre-

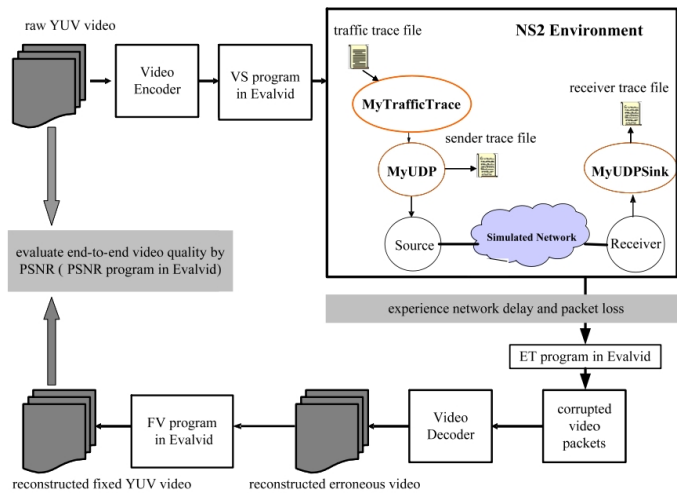


Fig. 2. Interfaces between EvalVid and NS-2[4].

spond to the decoded (YUV) frame with the same number. FV fixes this issue, by matching display (YUV) frames to transmission (coded) frames.

In[4], NS-2 has been integrated into Evalvid. By doing that, it has brought a huge libraries of network protocols and simulations into the video evaluation platform.

Fig. 2 shows interfaces between Evalvid and NS-2 which has been developed in [4]. Here, the sender trace file generated by VS will be used as the input for the NS-2 based simulation script. The MyTrafficTrace agent extracts the frame type and the frame size of the video trace file, fragments the video frames into smaller segments, and sends these segments to the lower UDP layer at the appropriate time according to the user settings specified in the simulation script file.

The original FV was only capable of fixing the difference between the sending and decoding orders. If there is a corrupted frame, FV can partly fix it by simply truncating it or replacing it with a null frame so that the decoder can still recognize the frame as a decodable one. However, when a decodable frame depends on a previous frame to be decoded (as in the case of B frames depending on I and P frames, P frames depending on I and previous P frames to be decoded) and that previous frame was corrupted, then FV cannot do anything. FV was also further developed to solve this problem.

III. SVC AND ITS RTP PAYLOAD

Scalable Video Coding was standardized as an extension of H.264/AVC. Deriving from H.264/AVC, it maintains the concepts of using a Video Coding Layer (VCL) and a Network Abstraction Layer (NAL) [2].

A. Video Coding Layer

In H.264/AVC, each video frame to be encoded will be partitioned into smaller coding units called macro-blocks[2]. A macro-block will cover a rectangular picture area of luminance samples. Not all macro-blocks are fully encoded, most of them can be spatially or temporally pre-

dicted before being fed into the VCL encoder. Outputs of the VCL are slices: a bit string that contains the macro-block data of an integer number of macro-blocks (making a full frame) which are normally organized into slices according to the frame scanning order; and the slice header (containing the spatial address of the first macro-block in the slice, the initial quantization parameter, and similar information)[10]. In both H.264/AVC and SVC, there are three main types of slices:

I slice: intra-picture coding using intra-spatial prediction from neighboring regions. This type of slice is self-contained and can be decoded without the reference to any other slice.

P slice: intra-picture predictive coding and inter-picture predictive coding with one *prediction* signal for each predicted region. This type of slice can only be decoded with reference information from previous I or P frame.

B slice: inter-picture bi-predictive coding with two prediction signals that are combined with a weighted average to form the region prediction. This type of slice can only be decoded with reference information from the previous and successive I or P frame.

B. Network Abstraction Layer

If the VCL is the interface between the encoder and the actual video frames, the Network Abstraction Layer (NAL) is the interface between that encoder and the actual network protocol, which will be used to transmit the encoded bit-stream. The NAL encoder encapsulates the slice output of the VCL encoder into Network Abstraction Layer Units (NALU), which are suitable for transmission over packet networks or used in packet oriented multiplex environments[11]. In order to generate proper NAL units, we must pre-define the network protocol that we want to use to transmit the video bit-stream. H.264/AVC and SVC support encapsulating VCL slices into a number of network protocol (H.320, MPEG-2, and RTP...)[12] in which RTP is mostly used because of its popularity.

SVC extended the H.264/AVC standard by providing scalability. There are three main kinds of scalability that SVC can support:

Temporal scalability: A bit-stream provides temporal scalability when the set of access units (a set of NAL units that always contains exactly one primary coded picture) can be partitioned into a temporal base layer and one or more temporal enhancement layer(s). A strictly requirement for a bit-stream to be called temporal scalable is that, when we remove all access units of all temporal enhancement layers with a temporal layer identifier higher than k ($1 < k < maxlayer$), then the remaining layers still form a valid bit-stream for a SVC decoder (when $k=1$, then we have a base-layer bit-stream which must be compatible with conventional H.264/AVC decoders). Due to its non-reference property, B slices are often used to form temporal enhancement layers.

Spatial scalability: A bit-stream contains of multiple

layers, in which each layer corresponds to a supported spatial resolution and can be referred to by a spatial layer with a *dependency identifier*. In each spatial layer, motion-compensated prediction and intra-prediction are employed as in single-layer video coding. However, among layers, an inter-layer prediction mechanisms are applied to improve the coding efficiency and rate-distortion efficiency by using as much lower layer's information as possible. Lower layer pictures do not need to be present in all access units making it possible to combine spatial and temporal scalability.

Quality (SNR) scalability: This scalability can be considered as a special case of spatial scalability with identical picture sizes of base and enhancement layers. Quality scalability comprises of *coarse-grain quality scalable* (CGS) coding, *medium-grain quality scalable* (MGS) coding and *fine-grain quality scalable* (FGS) coding. In CGS, inter-layer prediction is also used. A higher quantization step size will be provided by the enhancement layers to provide a better quality for the lower layers. However, this multi-layer concept for quality scalable coding only supports a few selected bit rates in a scalable bit stream. In general, the number of supported rate points is identical to the number of layers. Switching between different CGS layers can only be done at defined points in the bit stream. Furthermore, the multi-layer concept for quality scalable coding becomes less efficient, when the relative rate difference between successive CGS layers gets smaller. MGS provides a better coding efficiency for bit-streams that have to provide a variety of bit-rates. With MGS, any enhancement layer NAL unit can be discarded from a quality scalable bit stream and thus packet-based quality scalable coding is provided. *Fine-grain quality scalable* (FGS) provides a coding prediction structure mechanism that completely omits drift (the motion-compensated prediction loops at encoders and decoders are not synchronized because quality refinement packets are discarded from a bit-stream).

Combined scalability: In some cases, quality, spatial, and temporal scalability can be combined.

IV. EVALSVC

Our work manages to develop a video transmission evaluation framework supporting SVC's NALU extension types. The most difficult problem is that those extending types haven't been fully defined and standardized by IETF. However, it should be noticed that, the basic NALU extension types (e.g., types 14, 15, 20) have been spared for SVC extensions from AVC NALU types. So we are going to support only those NALU extensions in our EvalSVC framework since they have already reflected the main concepts of SVC. Other NALU types, such as Payload Content Scalability Information (PACSI), Empty NAL unit and the Non-Interleaved Multi-time Aggregation Packet (NI-MTAP), which are being drafted in[11], are out of our

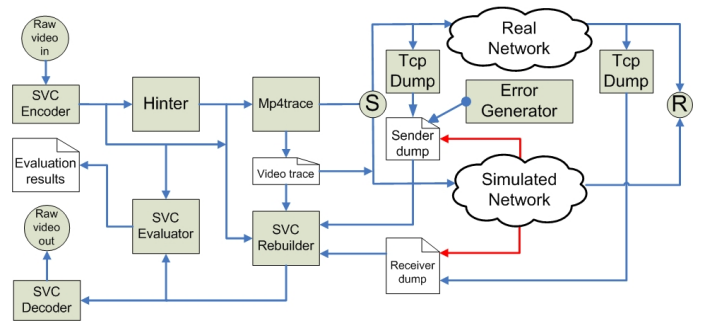


Fig. 3. EvalSVC's diagram.

0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
F	N	R	I	Type				R	I	PID				N	D	I	Q	I	T	U	D	O	R	R							

Fig. 4. SVC NALU's header.

scope.

A NAL unit comprises of a header and a payload. In AVC, the NALU's header is 1 byte length[13]. Meanwhile, a SVC's NAL header can be 1, 2, or 3 octet length[10]. The first octet of SVC's NAL header is identical with AVC (Fig. 4). It contains 3 fields of which 2 first fields (F, NRI) are spared for signaling wire-line/wireless gateway, and the importance of that NALU. The last field in the first octet of the SVC's NAL header is *NALU Type* specifying the NAL unit payload type. NAL unit type 14 is used for prefix NAL unit, NAL unit type 15 is used for subset sequence parameter set, and NAL unit type 20 is used for coded slice in scalable extension. NAL unit types 14 and 20 indicate the presence of three additional octets in the NAL unit header. NALU types 15 contents header information which is not necessary to be repeatedly transmitted for each sequence of of picture[14]. This sub-sequence parameter set can be transmitted on an "out-of-band" transmission for error resilience. We will need this information about the NALU types when we reconstruct the possibly corrupted SVC bit-stream at the receiver side. PRID (priority ID) specifies a priority identifier for the NALU. A lower PRID indicates a higher priority. DID (dependence ID) indicates the inter-layer coding level of a layer representation. QID (quality ID) indicates the quality level of an MGS layer representation. TID (temporal ID) indicates the temporal level of a layer representation.

Fig. 3 illustrates basic components of our EvalSVC platform. Some external tools are also integrated into EvalSVC to support the data-flow of the entire framework.

Raw video in : This is the input video. Normally the YUV or CIF formats are used as they are acceptable by SVC encoders as well as common video capturing devices.

SVC encoder/decoder : We use JSVM[15] as our main SVC codec.

Hinter : This component is derived from the mp4box tool of the GPAC library[7]. The main role of this component is to packetize SVC's NALU into RTP packets and add a hint track to the SVC bit-stream. We can consider the hint track as an in-band signal-

ing for the SVC bit-stream. Another option is to distribute the hint track in the format of a SDP file via a separate channel as out-band signaling.

Mp4trace : This component acts as a video sender. Its main part is to send the hinted SVC bit-stream out to the network using the packetization information it has from the Hinter. It also logs the sequence numbers, types, and sizes of the video frames, and the number of UDP packets used to transmit each frame (since the frame size may exceed the UDP/RTP maximum payload sizes), and its sending timescale. Mp4trace can work in streaming mode or camera mode.

Networks : 2 kinds of networks can be used in EvalSVC, real and simulated networks. Real network's conditions can be obtained by using real IP connections over the Internet. Tcpcap can be used to trace the real network traffic at both ends and to form the sender's and receiver's dumping files. We can also use NS-2 simulated network to form the sender's and receiver's dumping files. Using a NS-2 based simulation network, one can test a new SVC video transmission algorithm, or evaluate the performance of SVC video transmission over a conventional network model (supported by NS-2). A simulated network can comprise of many relaying nodes. Since the SVC bit-stream comprises of multiple layers, enhancement layers can be discarded at the relaying nodes according to the simulation scripts.

SVC Re-builder : Being the heart of EvalSVC, the Re-builder will collect all data from sender's, receiver's dumpings and video trace files, take both the SVC encoded bit-stream and the hinted file at the sender into account and reconstruct a possibly-corrupted output SVC bit-stream at the receiver. The SVC re-builder must understand SVC NALU headers in order to properly rebuild the corrupted SVC bit-stream. When encountering a missing packet, or a missing frame, the SVC re-builder has two options. It can truncate the SVC video frame or fill that frame with zero (or a default value). Other QoS measurements of the network such as end-to-end delay, jitter, loss rate, sender's and receiver's bit-rate will also be generated.

Error Generator : Normally, an optimal transmission condition can be obtained by using a direct connection between a sender and a receiver. We can use the Error Generator to modify the dump and trace files according to a pre-defined error distribution function.

SVC Evaluator : This component will compare the bit-stream from the output of the SVC Re-builder with the original bit-stream from the sender. Objective and subjective quality evaluation (PSNR, MOS) of the SVC video transmission will be carried out at this component.

Sender/Receiver nodes : Real or simulated nodes on the transmission network. They are the departure and destination of the video transmission.

A sample evaluation session using EvalSVC starts with the raw video taken from a file or real-time captured by

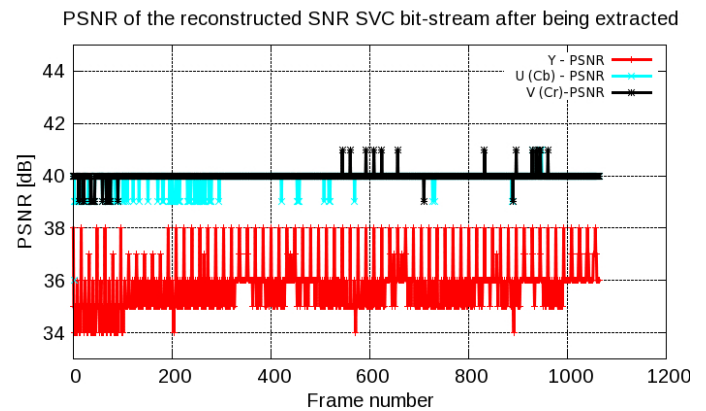


Fig. 5. PSNR of the reconstructed SNR SVC bit-stream after being extracted.

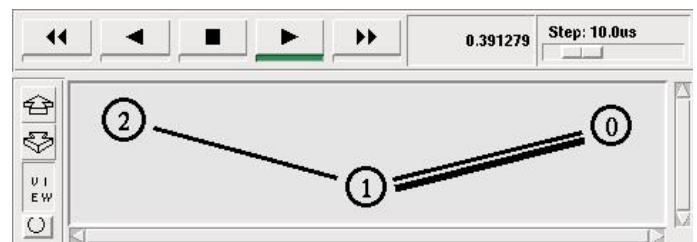


Fig. 6. NS-2 based simulation diagram of video transmission over a bottleneck network.

a camera. This raw video will then be encoded by the SVC encoder to form a SVC bit-stream. The SVC encoded bit-stream is fed into the Hinter to be packetized into RTP packets. A hint track will also be added to the original bit-stream. Mp4trace will send the hinted file (using streaming or camera mode) from the Sender node to the real/simulated network. A video trace file, a sender and a receiver dumping files will be generated. Using information from all of these files, and the original bit-stream, the SVC Re-builder will reconstruct the received bit-stream and feed it to the SVC Evaluator for generating the video transmission results. The reconstructed SVC video can also be delivered to the SVC decoder to get the output video play-out at the receiver side. Fig. 5 shows the PSNR of the extracted, decoded SVC bit-stream. For example, a cif-size raw file with 1065 frames is encoded using SNR SVC. The output bit-stream is sent via a real direct IP connection from a sender to a receiver. We manually generate errors by erasing entries at the sender's and receiver's dumping files. At the receiver, the received bit-stream is re-constructed by using the re-builder component of EvalSVC. Since the JSVM decoder cannot decode a corrupted bit-stream, we need to extract the uncorrupted base-layers out of the corrupted bit-stream for it to be decoded by the decoder. Fig. 5 shows the PSNR of all 1065 frames of the sample SNR SVC video. From this PSNR result, we can get the MOS value of the reconstructed SNR SVC at the receiver which is equal to 4.89 (between Good and Excellent).

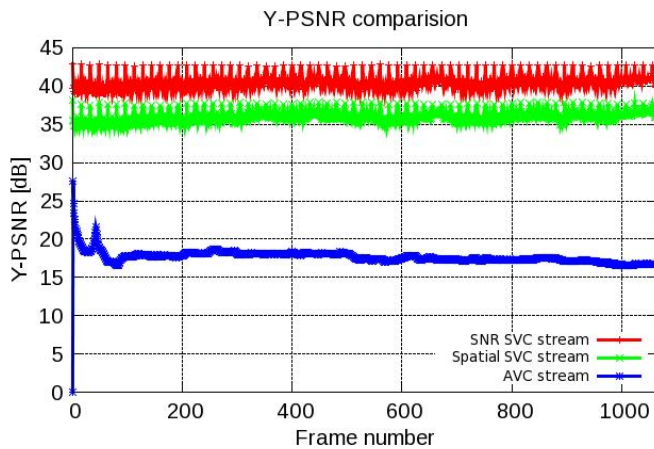


Fig. 7. Y-PSNR comparison among AVC, SNR SVC and Spatial SVC streams.

We can also use EvalSVC to evaluate the transmission of different kinds of SVC streams on a simulated network using NS-2. In the first simulation scenario (Fig. 6), we try to find out the best SVC method which can afford the most with the bottleneck condition of the network. To simulate the bottleneck condition, 3 nodes are built using NS-2: node 0 (the sender), node 1 (the relay), and node 2 (the receiver). The first link (link 1), connecting node 0 and node 1, has a bandwidth of 400 kbps, 1 ms delay. The second link (link 2), connecting node 1 and node 2 has a bandwidth of 100 Mbps, 1 ms delay. This network configuration will create a bottleneck on link 1. Firstly, a CIF-size AVC stream is sent from node 0 to node 2 via node 1. In the second and third simulations, a SNR SVC stream and a Spatial SVC stream (both CIF-size) are sent respectively via the same route from node 0 to node 2. We do not use the temporal SVC in our simulation since a temporal SVC stream is identical with an AVC stream. We use EvalSVC to evaluate the Y-PSNR performance of these 3 streams. Fig. 7 shows that, when bottom-neck occurs, SNR SVC has the best and AVC has the worst Y-PSNR performance. MOV grades of AVC, spatial SVC, and SNR SVC streams are 1.02, 4.07, and 5, respectively. We can conclude that, AVC is very sensible to bottleneck, a single bottleneck in the transmission route can easily block the entire communication session. Meanwhile, all SVC streams can afford quite well ($MOS > 4$) with the bottleneck condition of the network, among those, SNR SVC has the best performance.

V. CONCLUSION AND FUTURE WORK

In this paper, we have introduced EvalSVC, our evaluation platform for Scalable Video Coding video transmission. The main purpose of this work is to fill the gap between the design, evaluation and implementation processes of variable visual services based on Scalable Video Coding. Using our newly developed framework, we found that, SVC can afford better than AVC in bottleneck conditions of the network. Among SVC, through simulation results, we found that, SNR SVC can afford the most against

the bottleneck problem of the network. Our future work is to apply this platform to evaluate our innovative multimedia transmission algorithms based on Scalable Video Coding. This tool-set is publicly available at[16].

REFERENCES

- [1] J. V. Team, "Advanced video coding for generic audiovisual services," *ITU-T Rec. H*, vol. 264, pp. 14496–10.
- [2] 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*, vol. 17, no. 9, pp. 1103–1120, 2007.
- [3] J. Klaue, B. Rathke, and A. Wolisz, "Evalvid-a framework for video transmission and quality evaluation," *Lecture notes in computer science*, pp. 255–272, 2003.
- [4] C. H. Ke, C. K. Shieh, W. S. Hwang, and A. Ziviani, "An evaluation framework for more realistic simulations of MPEG video transmission," *Journal of Information Science and Engineering*, vol. 24, no. 2, pp. 425–440, 2008.
- [5] S. Wolf and M. Pinson, "Video quality measurement techniques," 2002., 2002.
- [6] D. Wu, Y. T. Hou, W. Zhu, H. J. Lee, T. Chiang, Y. Q. Zhang, and H. J. Chao, "On end-to-end architecture for transporting MPEG-4 video over the Internet," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 10, no. 6, pp. 923–941, 2000.
- [7] J. L. Feuvre, C. Concolato, and J. C. Moissinac, "GPAC: open source multimedia framework," in *MULTIMEDIA'07: Proceedings of the 15th international conference on Multimedia*, 2007.
- [8] S. Wenger, M. M. Hannuksela, T. Stockhammer, M. Westerlund, and D. Singer, "RTP payload format for H. 264 video," *IETF RFC3984*, February, 2005.
- [9] I. Rec, "P. 800: Methods for subjective determination of transmission quality," *International Telecommunication Union*, 1996.
- [10] S. Wenger, Y. Wang, and M. M. Hannuksela, "RTP payload format for H. 264/SVC scalable video coding," *Journal of Zhejiang University-SCIENCE A*, vol. 7, no. 5, pp. 657–667, 2006.
- [11] S. Wenger, Y. K. Wang, T. Schierl, and A. Eleftheriadis, "RTP payload format for SVC video," *draft, Internet Engineering Task Force (IETF)*, September 2009.
- [12] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H. 264/AVC video coding standard," *IEEE Transactions on circuits and systems for video technology*, vol. 13, no. 7, pp. 560–576, 2003.
- [13] S. Wenger, A. G. Teles, and G. Berlin, "H. 264/avc over ip," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 645–656, 2003.
- [14] Y. Wang, M. M. Hannuksela, S. Pateux, A. Eleftheriadis, and S. Wenger, "System and transport interface of SVC," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 9, pp. 149, 2007.
- [15] J. Reichel, H. Schwarz, and M. Wien, "Joint scalable video model JSVM-8," *ISO/IEC JTC1/SC29/WG11 and ITU-T SG16 Q. 6, JVT-U*, 2006.
- [16] Tien A. Le, Quang H. Nguyen, and Anh M. Nguyen, *EvalSVC tool-set: <http://code.google.com/p/evalsvc/>*, 2009.