myEvalSVC: an Integrated Simulation Framework for Evaluation of H.264/SVC Transmission

Chih-Heng Ke
Department of Computer Science and Information Engineering, National Quemoy University
Kinmen, Taiwan
[e-mail: smallko@gmail.com]
*Corresponding author: Chih-Heng Ke

Received October 6, 2011; revised December 26, 2011; accepted January 8, 2012;
Published January 31, 2012

Abstract

The ever-increasing demand for H.264 scalable video coding (H.264/SVC) distribution motivates researchers to devise ways to enhance the quality of video delivered on the Internet. Furthermore, researchers and practitioners in general depend on computer simulators to analyze or evaluate their designed network architecture or proposed protocols. Therefore, a complete toolset, which is called myEvalSVC, for evaluating the delivered quality of H.264/SVC transmissions in a simulated environment is proposed to help the network and video coding research communities. The toolset is based on the H.264 Scalable Video coding streaming Evaluation Framework (SVEF) and extended to connect to the NS2 simulator. With this combination, people who work on video coding can simulate the effects of a more realistic network on video sequences resulting from their coding schemes, while people who work on network technology can evaluate the impact of real video streams on the proposed network architecture or protocols. To demonstrate the usefulness of the proposed new toolset, examples of H.264/SVC transmissions over 802.11 and 802.11e are provided.

Keywords: H.264/SVC, SVEF, NS2, myEvalSVC, network simulation

DOI: 10.3837/tiis.2012.01.021

1. Introduction

H.264/MPEG10 or advanced video coding (AVC) [1][2] is an industrial video compression standard, which converts digital video into a format that consumes less capacity for storage or transmission [3][4][5]. Video recording, video playback, video surveillance, and video conferencing are common examples of applications of H.264/AVC technology. However, the limitation of scalability restricts the ability of H.264/AVC to meet different needs of different users with different displays connected through different network links. Therefore, H.264 scalable video coding (H.264/SVC) [6] was proposed to overcome the limitation. Spatial scalability, temporal scalability, and signal-noise-ratio (SNR) scalability are three important features of the H.264/SVC standard. Spatial scalability can provide the standard with the ability to adapt to the video's spatial resolution. Temporal scalability enables adaptation to the frame rate, and SNR scalability enables adaptation to the video quality.

SVC contains one base layer and one or more enhancement layers. The base layer provides the basic quality of video. Moreover, in order to be backward compatible, the base layer must be recognized by all conventional H.264 decoders. Adding the enhancement layer to the base layer increases the video quality. On the other hand, when the available bandwidth is insufficient, dropping one or more enhancement layers, partially or completely, is an easy way to avoid run-time video transcoding.

For delivering better H.264 video quality, researchers or practitioners usually need an experimental environment to test their ideas. Traditionally, they implement the video server, set up the network, and build the video client. Then, they use the H.264 video traffic to transmit over the testbed and measure the results [7][8]. These steps are time consuming and expensive. Furthermore, if one interesting result is obtained, it is not easy to repeat the experiment to find the factors that make the interesting result to appear again. A simulation-based method is another way to test an idea. References [9][10][11][12] provide examples that use simulations to carry out the evaluations. However, almost none of these works provide their evaluation framework in the public domain.

To the best of my knowledge, EvalSVC [13] is the only publicly available toolset to perform quality evaluation of delivered H.264/SVC video in a network simulation environment. However, EvalSVC does not clearly indicate how it handles missing or corrupted Network Abstraction Layer Unit (NALU) losses and how it handles play-out delay constraints. The current Joint Scalable Video Model (JSVM) codec (version 9.19), which is the existing reference open source software for H.264/SVC coding and decoding, cannot decode properly in these situations. In addition, EvalSVC does not provide any simulation example and explanation on how to use it. For providing a more realistic simulation environment, an H.264/SVC transmission evaluation framework, which is called myEvalSVC, is proposed in this paper. This framework integrates the H.264 Scalable Video coding streaming Evaluation Framework (SVEF) [14][15] with the NS2 [16], which is a widely adopted network simulator. By using this combination, people who work on video coding can simulate the effects of a more realistic network on video sequences resulting from their coding schemes, while people who work on network technology can evaluate the impact of real video streams on the proposed network architecture or protocols. In fact, it is very difficult for beginners to set up the whole simulation environment. Microsoft Visual C++ [17] is needed to build the JSVM; Python [18], to run the SVEF; and Cygwin [19], to build NS2. Based on the virtual machine technology, a VirtualBox [20] guest operating system image is available at

[21]. Beginners just need to download the image and install VirtualBox software. They can start using myEvalSVC immediately. In [21], examples of H.264/SVC transmission over 802.11 and 802.11e are provided. A user can start by encoding the raw video, parse the compressed video content, send the corresponding packets at the assigned time, and then perform delivered video quality evaluation after simulations. In [22], a sender transmits one base layer and two enhancement layers of packets to three receivers. One receiver receives all three layers of packets, another receives one base layer and one enhancement layer of packets, and the other only receives the base layer of packets. This example shows the benefits of multicasting H.264/SVC to users with different needs. All steps are well explained and illustrated to make it easy for beginners to start using myEvalSVC. Moreover, the concept to combine SVEF with different network simulators is also presented at [21]. It will help the potential users in handling similar integration tasks.

The remainder of this paper is organized as follows. Section 2 provides an overview of video coding, an evaluation of video transmission methods, and a description of the SVEF. Section 3 describes the proposed myEvalSVC evaluation framework. Examples of usage of the framework in IEEE 802.11 and 802.11e wireless networks are given in section 4. Finally, section 5 presents the concluding remarks.

2. Background and Related Work

2.1 Overview of VIdeo Coding

Non-scalable video coding: There are three basic types for Moving Picture Experts Group (MPEG) video frames: (1) I-frame, or intra-coded frame, where the frame is encoded independently of other frames and decoded by itself, (2) P-frame, or predictive-frame, where the frame is encoded using predictions from a preceding I- or P-frame in the video sequence, and (3) B-frame, or bi-directionally predictive-coded frame, where the frame is encoded using predictions from preceding and succeeding I- or P-frames.

Generally, the entire video sequence can be decomposed into smaller units, which are then coded together, called the Group of Pictures (GOP). A GOP pattern is characterized by two parameters, G (N, M): N is the I-to-I frame distance and M is the I-to-P frame distance. For example, as shown in **Fig. 1**, G (9, 3) means that the GOP includes one I-frame, two P-frames, and six B-frames. The second I-frame shown in **Fig. 1** indicates the beginning of the next GOP. The arrows indicate that the B-frames and P-frames decoded are dependent on the preceding or succeeding I- or P-frames.

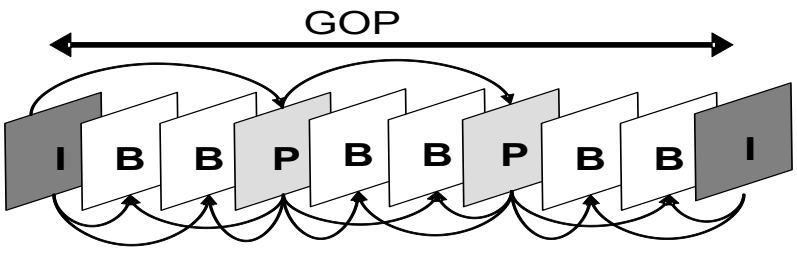

Fig. 1. An example of MPEG coding with GOP (N = 9, M = 3)

Scalable video coding (SVC): In scalable or layered video coding, the video is encoded

hierarchically into a base layer and one or more enhancement layers. Decoding the base layer offers low but standard video quality, while decoding the base layer together with additional enhancement layers provides further refinement of the video quality. There are different forms of scalability, including temporal, spatial, and SNR scalability.

Fig. 2 shows an example of the temporal scalable encoding. The I- and P-frames form the base layer, and the B-frames form the enhancement layer. The base layer provides the basic video quality with a lower frame rate. Adding the enhancement layer to the base layer increases the smoothness of the video quality.

H.264/SVC is a scalable extension of H.264/AVC. It is a current standardization of the Joint Video Team (JVT). An encoded SVC bitstream consists of an H.264/AVC-compatible base layer and one or more scalable enhancement layers. Conceptually, the design of H.264/AVC covers a Video Coding Layer (VCL) and a Network Abstraction Layer (NAL). While the VCL creates a coded representation of the source content, the NAL formats these data and provides the header information in a way that enables simple and effective customization of the use of VCL data for a wide variety of systems.

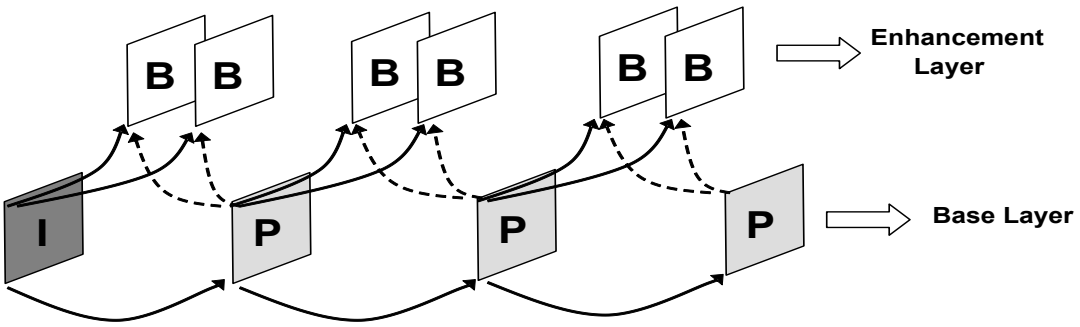

Fig. 2. An example of temporal video coding

Multiple description video coding: Multiple description coding (MDC) [23] arose in connection with communicating speech over the telephone network. The idea was to split the information from a call into two parts that are sent on two different paths. In normal operation, two parts are received and combined for usual voice quality. However, an outage of one link or other can still be accommodated by reducing the voice quality. This idea of channel splitting inspired the so-called multiple description video coding.

In multiple description video coding, the video signal is split into multiple sub-streams, where each of the sub-streams is decodable in a stand-alone fashion. The more sub-streams are received; the more information of the original source can be restored. Note that these properties are in contrast to the scalable or layered video coding schemes, in which the enhancement layer(s) would become useless for the receiver if the base layer is lost.

2.2 Evaluation of Video Transmission

Generally, there are three different ways to evaluate video transmission.

Using real bit streams: This method uses the actual output of video encoding for video transmission evaluation. *RealTracer* [24], a set of tools for measuring the performance of *RealVideo*, is an example. It includes *RealTracker*, a customized video player that can play streaming *RealVideo* clips and record system performance statistics and user ratings, along with *RealData*, a data analysis tool that helps manage, parse, and analyze statistical data captured by *RealTracker*. One advantage of this kind of method is that it allows the quality of

the video to be visually evaluated. The network level metrics, such as bandwidth usage, frame-rate, jitter etc, can also be obtained. However, this kind of tool focuses mainly on real networks. This may prevent networking people from evaluating their proposed protocols in a timely manner, because they commonly use simulation tools to verify the effectiveness of their designs before deploying the protocols in real networks. In order to solve the above-mentioned problem, an evaluation framework, which is called myEvalvid, was proposed for more realistic simulations of MPEG video transmission [25][26]. myEvalvid, which combines Evalvid [27] and NS2, allows researchers and practitioners analyze the performance of real video streams through simulations under a large range of network scenarios. However, the myEvalvid only supports not scalable video coding. Besides, the user needs to install NS2 simulators first, and then follow the instructions provided in [26] to manually setup related files in NS2. This is not an easy task, especially for NS2 beginners. As a consequence, myEvalSVC is provided in VirtualBox operating system image. Users can be quickly familiar with this integrated evaluation framework by running examples and then start their own research work. Furthermore, myEvalSVC not only supports non-scalable video coding but also scalable video coding.

Using traffic traces: The video traffic trace is an abstraction of real video stream. It typically gives the frame number, frame type, and frame size in a text file to describe the characteristics of real video traffic. Reference [28] indicates a good website that provides many kinds of video traffic traces, such as H.264, MPEG, or MDC traces. The advantage of using traffic traces is that one does not need to be concerned about copyright issues, because they do not contain the actual video information. Nevertheless, for a simulation study, usually only network level metrics can be obtained. In the case of evaluating video transmission, network level metrics may be insufficient to rate the quality perceived by an end user. Take the loss rate as an example: relatively low loss rates do not necessarily mean good delivered video quality. A 3% packet loss percentage could translate into a 30% frame error probability. Modern video codecs are hierarchical, so the loss of the I-frame would cause other frames in the same GOP become useless. Furthermore, it is hard to study the effects of proposed network mechanisms on different characteristics of the same video extensively, because the encoding settings for the publicly available video traffic traces are limited.

Using video traffic models: A video model captures the properties of real video bit streams in a mathematical way. This method is typically developed based on the statistical properties of a set of video trace samples of real video traffic. Transform Expand Sample (TES) [29] is an example of this kind of methodology for generating data that closely match (in terms of its marginal distribution and auto-correlation function) any set of given observation of a time series. The developed model can be used for the mathematical analysis of networks, but it lacks the possibility of visualizing a transmitted video.

2.3 The SVEF Evaluation Framework

The structure of the SVEF evaluation framework is shown in **Fig. 3**, redrawn from [9]. The main components of the evaluation framework are described as follows:

Raw YUV video: This is the video source file. These files are commonly in the YUV 4CIF (704 × 576), YUV CIF (352 × 288), or QCIF (176 × 144) formats.

JSVM Encoder: The encoding process is based on configuration files. Users can enable spatial scalability, temporal scalability, SNR scalability, or combined scalability. **Table 1** shows an example of encoding process. The second field shows the frame number and frame type. The third field is in temporal_id (TId), dependency_id (DId), and quality_id (QId) format. DId allows spatial scalability, TId denotes the temporal scalability, and QId represents

the quality scalability. For the current version 1.4, the SVEF does not take the spatial scalability into consideration, and only supports SVC with a single dependency layer and an arbitrary number of quality enhancement layers. Therefore, with the same value for the DId and TId parameters, a NALU having "qid (the value of QId) > 0" depends on NALUs having "qid − 1". With the same value for the DId and QId, a NALU having "tid (the value of TId) > 0" and "qid=0" depends on NALUs having "tid − 1". The remaining fields indicate the quantization parameter, Y-PSNR, U-PSNR, V-PSNR, and encoded frame size. The Peak Signal Noise Ratio (PSNR) can be calculated for both luminance (Y-PSNR) and chrominance (U-PSNR and V-PSNR) components of the video. Since the human eye is more sensitive to luminance (brightness) than chrominance (colour), the PSNR is typically evaluted only for the luminace (Y) component. The following equation shows the definition of the PSNR between the luminance component Y of source image and destination image D:

$$PSNR(n)_{dB} = 20 \log_{10} \left(\frac{V_{peak}}{\sqrt{\frac{1}{N_{col} N_{row}} \sum_{i=0}^{N_{col}} \sum_{j=0}^{N_{row}} [Y_S(n,i,j) - Y_D(n,i,j)]^2}} \right) \quad (1)$$

where $V_{peak} = 2^k$-1 and k = number of bits per pixel. N_{col} presents the number of columns, while N_{row} is the number of rows in an image. PSNR measures the error between a reconstructed image and the original one. A larger PSNR value corresponds to a better image quality. For more detailed information, please refer to the JSVM Software Manual [30].

Table 1. Example of encoding process

AU	0: I	T0 L0 Q0	QP 29	Y 37.2503	U 40.7950	V 43.6207	51944 bit
AU	4: I	T0 L0 Q0	QP 29	Y 37.2076	U 40.8874	V 43.6978	52992 bit
AU	2: B	T1 L0 Q0	QP 33	Y 36.4711	U 40.7747	V 43.6246	5888 bit
AU	1: B	T2 L0 Q0	QP 34	Y 36.5085	U 40.7194	V 43.5171	2656 bit
.....							

JSVM BitStreamExtractor: After encoding, a H.264 video file is generated. This video file is then fed into BitStreamExtractor to produce the original Network Abstraction Layer Unit (NALU) trace file. However, this trace file does not contain frame number information. So this trace file is processed by an F-N Stamp to generate a NALU trace with frame number information in it. In **Table 2**, the meanings of all fields are as follows: memory offset, NALU-size, DId, TId, UId, Type, Discardable, Truncatable, Frame-number, and Frame-sending time or Frame-receiving time.

Table 2. Example of the original NALU Trace

0x00000000	97	0	0	0	StreamHeader	No	No	-1	0
0x00000061	13	0	0	0	ParameterSet	No	No	-1	0
0x0000006e	8	0	0	0	ParameterSet	No	No	-1	0
0x00000076	18	0	0	0	SliceData	No	No	0	0
0x00000088	6484	0	0	0	SliceData	No	No	0	0
.....									

Streamer: The streamer reads the original NALU trace file, loads the data from H.264 file,

and then sends the NALUs over the IP network. The sent out packets consists of an IP header, UDP header, custom layer-5 header, and then payload. If a packet is too large and exceeds the fragmentation limit, the SVC will let the IP layer to do IP fragmentation/reassembly jobs.

MiddleBox: This component is optional. The creators of the SVEF use this MiddleBox as an example to do packet scheduling. When the available bandwidth is less than the sending rate, MiddleBox will decide which packets can send out and which packets cannot in accordance with the DId, TId, and QId fields in the packet header.

The Receiver-side Tools (NALU-Receiver, NALU-Filter, and Frame-Filler): In the receiver side, the NALU-Receiver is used to receive packets, and builds a received NALU trace file at the same time. The file format is the same as in **Table 2**, but the last field is recorded as frame-receiving time. Next, the received NALU trace file is processed by the NALU-Filter. This filter reorders the NALUs in accordance with the sending order, and removes NALUs that are too late or the NALUs with unfulfilled decoding dependencies. Then, the filtered NALU trace file is passed to JSVM BitStreamExtractor to retrieve the NALUs that are effectively decoded at the receiving side, and then decodes them into YUV video. It is worth noting that JSVM decoder does not directly decode the received NALUs. This is because the JSVM decoder cannot handle out-of-order, corrupted, or missing NALUs properly. In the final step, in order to compare the PSNR values, the same number of frames with the original raw YUV video is needed. Therefore, Frame-Filler is used to conceal the missing frames by copying the previous frame.

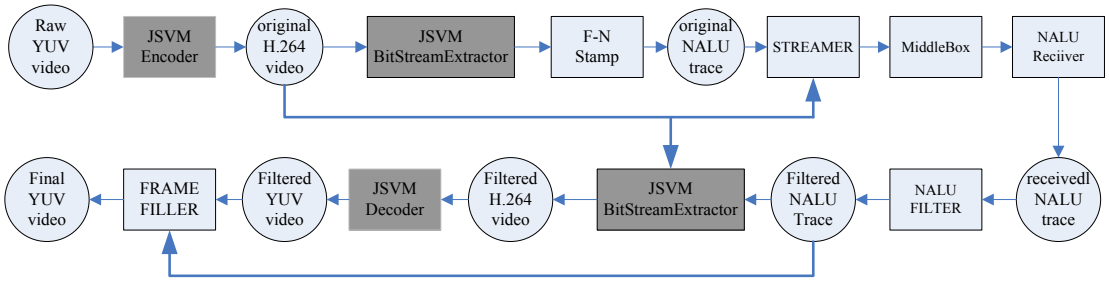

Fig. 3. SVEF software chain

3. The myEvalSVC Evaluation Framework

Fig. 4 shows the myEvalSVC evaluation framework. It is mainly based on the SVEF evaluation, and extended by three trace file converting programs (prepare_ns2sendtrace, prepare_receivedtrace1, and prepare_receivedtrace2), two network-level performance evaluation tools (Pe2edelay and PLossRate), and three connecting NS2 agents (myEvalSVC, MyUDP, and myEvalSVC_Sink).

After encoding the raw YUV video and content analysis, the original NALU trace is passed to the prepare_ns2sendtrace program to generate the NS2 traffic trace file. This file is the input of the myEvalSVC agent and contains the sending time, frame size, DId, TId, QId, and the number of fragmented packet fields for each record. The myEvalSVC agent reads each record from the NS2 traffic trace file, generates the corresponding number of packets, and sends them to the lower UDP layer at the appropriate time according to the user settings specified in the simulation script file. MyUDP is an extension of the UDP agent. It allows the user to specify the output file name of the sender trace file and records the sending time, packet ID, and the

packet size of each transmitted packet. Then, those packets go down to the lower layers and to the simulated networks. Researchers can design different protocols or evaluate the H.264/SVC over different network topologies easily by means of setting different parameters in the simulation script file. At the receiving side, the myEvalSVC_Sink agent is used to receive packets and record the corresponding receiving time, frame number, packet size, DId, TId, QId, packet ID, and the sending time in the user-specified receiver trace file.

After simulation, the number of records in the sender trace file and that in the receiving trace file are used to calculate the number of lost packets during transmission. By dividing the number of lost packets by the number of all sending packets, we get the packet loss rate. This is what PLossRate does. Moreover, if the sending time is subtracted from the receiving time for the same packet ID, the packet end-to-end delay can be obtained. Then, the received trace file is first processed by the prepare_receivedtrace1 program to obtain a frame-level received trace file. The same frame number packets are merged into one record and the last received packet is assigned to the frame receiving time. This frame-level received trace file, the NS2 traffic trace file, and the original NALU trace are further processed by prepare_received2trace to generate the received NALU trace needed by the SVEF evaluation framework. At the final stage, through NALU filtering, decoding, and frame filling, the final YUV video is produced. This final YUV video can compare with the raw YUV video to obtain frame-level PSNR for evaluating the end-to-end delivered video quality.

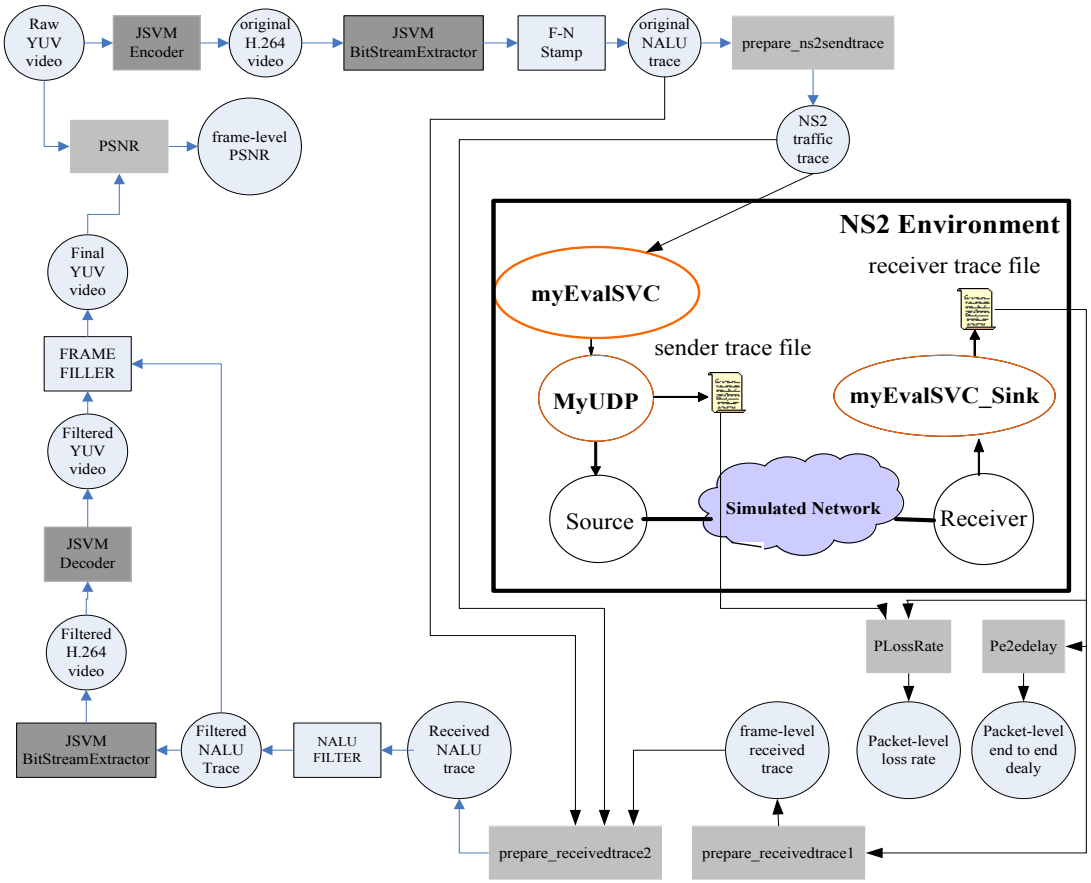

Fig. 4. The myEvalSVC evaluation framework

4. Usage Examples

To demonstrate the usefulness of the proposed myEvalSVC evaluation framework, examples of H.264/SVC transmissions over IEEE 802.11 [31] and IEEE 802.11e [32] networks are provided (see **Fig. 5**) in this section. The test video source, Foreman, used in the simulation is in YUV CIF (352 × 288) format and comprises 300 frames. It is encoded by JSVM (version 9.19) with only temporal scalability enabled. The resulting video parameters are summarized in **Table 3**.

Table 3. Parameters for Foreman video

Layer	Resolution	Frame rate	Bit rate	(DId, TId, QId)
0	352 × 288	7.5	514.10	(0,0,0)
1	352 × 288	15.0	548.70	(0,1,0)
2	352 × 288	30.0	588.20	(0,2,0)

The simulated scenario consists of three wireless nodes, i.e., n0, n1, and n2, and each node is within another node's transmission range. In the 802.11 case, n0 transmits H.264/SVC, CBR flow 1, FTP, and CBR flow 2, to n1, n1, n2, and n2 respectively as depicted in **Fig. 5-(a)**. In 802.11e cases, the H.264/SVC packets are mapped to the AC_VI (video) queue, the CBR Flow 1 and FTP packets are mapped to AC_BE (best effort) queue, and the CBR Flow 2 are mapped to AC_BK (background) queue (see **Fig. 5-(b)**). The other parameter settings are shown in **Table 4**.

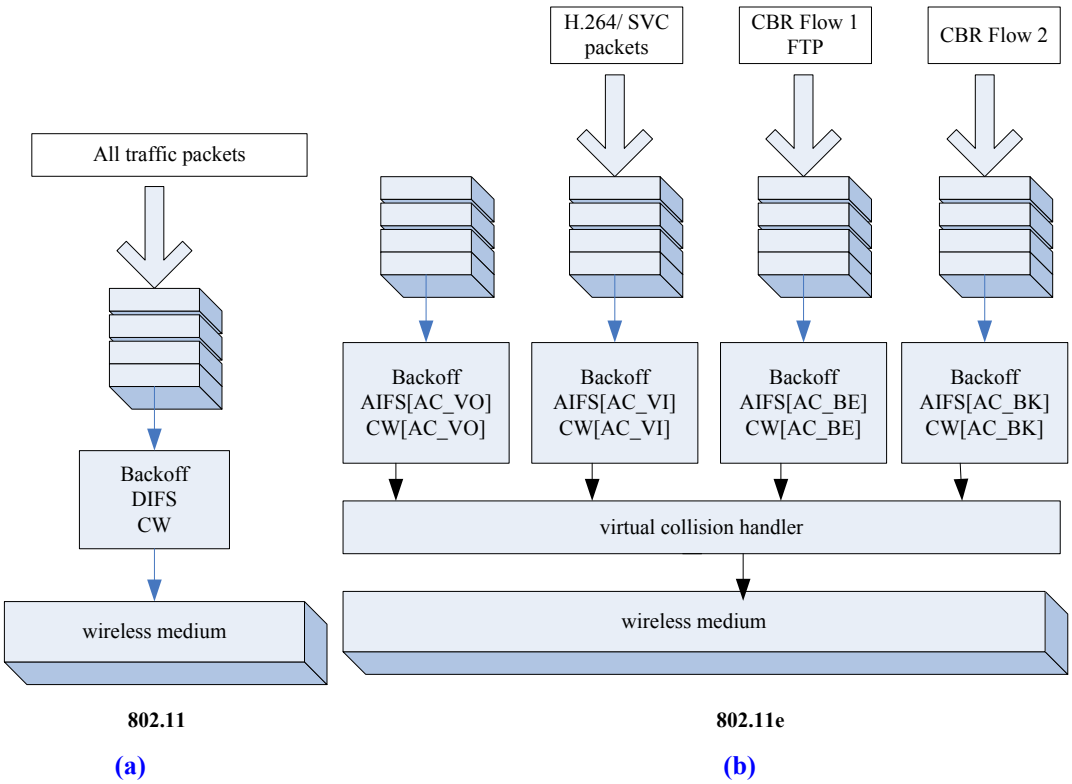

Fig. 5. Different H.264/SVC transmission schemes

Table 4. Simulation parameters

SIFS (μs)	10
Time slot (μs)	20
DIFS (μs)	50
CWmin	32
CWmax	1024
Physical header (bits)	192
MAC header (bits)	224
ACK (bits)	112
Data rate (Mbps)	1
Basic rate (Mbps)	1
Sending rate of CBR flow 1 (Mbps)	0.2
Sending rate of CBR flow 2 (Mbps)	0.3
Play-out delay (seconds)	5

Table 5. Packet Loss Rate

802.11	802.11e
51.44%	9.22%

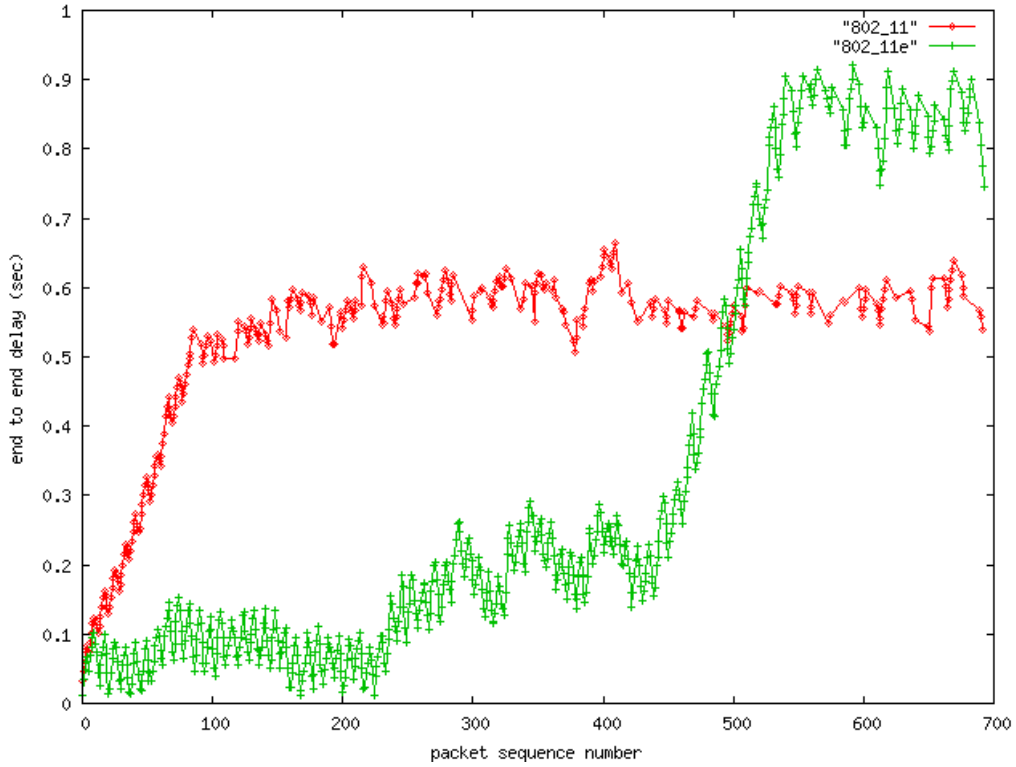

Fig. 6. End-to-End Delay

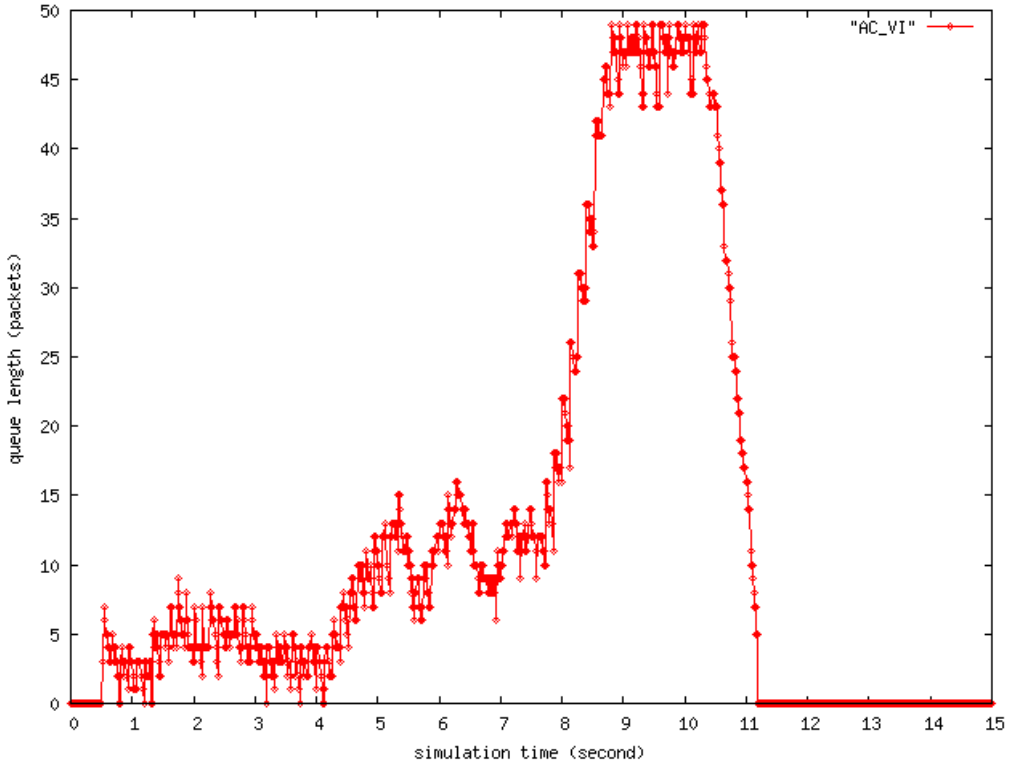

Fig. 7. Variation in AC_VI queue length in 802.11e

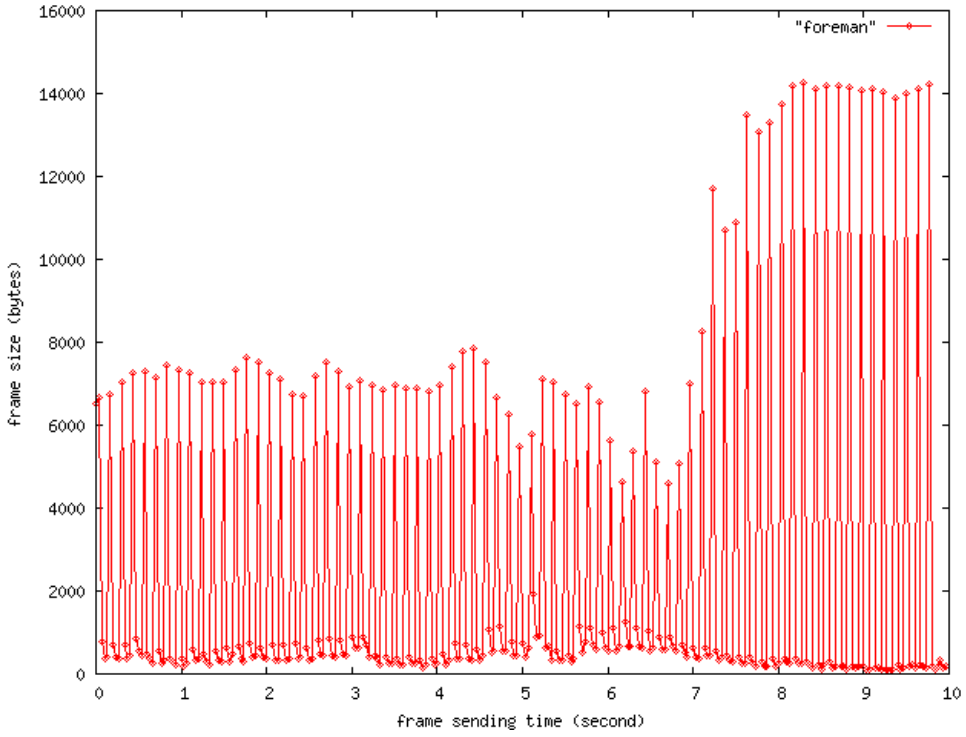

Fig. 8. The frame size versus frame sending time for Foreman video

Table 5 and **Fig. 6** show the network level performance metrics, i.e., the packet loss rate and end-to-end delay. It is clearly seen from **Table 5** that when all traffic packets are transmitted over 802.11 networks, the packet loss rate is high. This is because all packets go into the same output interface queue and the queue size is limited. When the queue is full, it starts to drop the packets. On the contrary, when video packets are transmitted over 802.11e, these packets do not need to contend with best effort or background traffic packets. Therefore, 802.11e can achieve the lowest packet loss rate. Next, if we compare the end-to-end delay, we can see that when the packet sequence number is below 500, the packets with 802.11e are lower than those with 802.11. However, when the packet sequence number is above 500, the packets with 802.11e become larger. This phenomenon can be explained by **Fig. 7** and **Fig. 8**. When the simulation time is below 8 seconds, the queue length is small. The video packets do not need to wait for so long. Therefore, the end-to-end delay is also small. However, when the simulation time is above 8 seconds, the AC_VI queue is augmented sharply. This is because the frames with TId=0 after 7 seconds have larger sizes and will be fragmented into many small packets. Then these packets are sent to AC_VI queue back to back. Consequently, the video packets have to wait longer when transmitting to the wireless medium.

Fig. 9 shows the PSNR values for different sending methods. The top curve represents the ideal PSNR values with no NALU loss, the middle curve is for video transmission over 802.11e, and the bottom curve is over 802.11. The average values are 35.9, 24.0, and 15.8 respectively. The results show that the H.264/SVC transmission over 802.11e can achieve better PSNR values than the transmission over 802.11. In addition, to illustrate how the difference in performance is received by an end user, the corresponding visual effects are shown in **Fig. 10** by means of the YUV display tool, i.e., YUVviewer [33]. In **Fig. 10**, the frame number 190, 191, and 192 are snapshotted for different scenarios. The frames in all_layers and 802.11e are almost the same. However, due to the high packet loss rate of 802.11, the exact frames cannot be decoded. So the displayed frames are those that can be decoded previously.

With the aid of the proposed myEvalSVC framework, researchers can easily evaluate their proposed network protocols or architecture and then visually compare the performance.

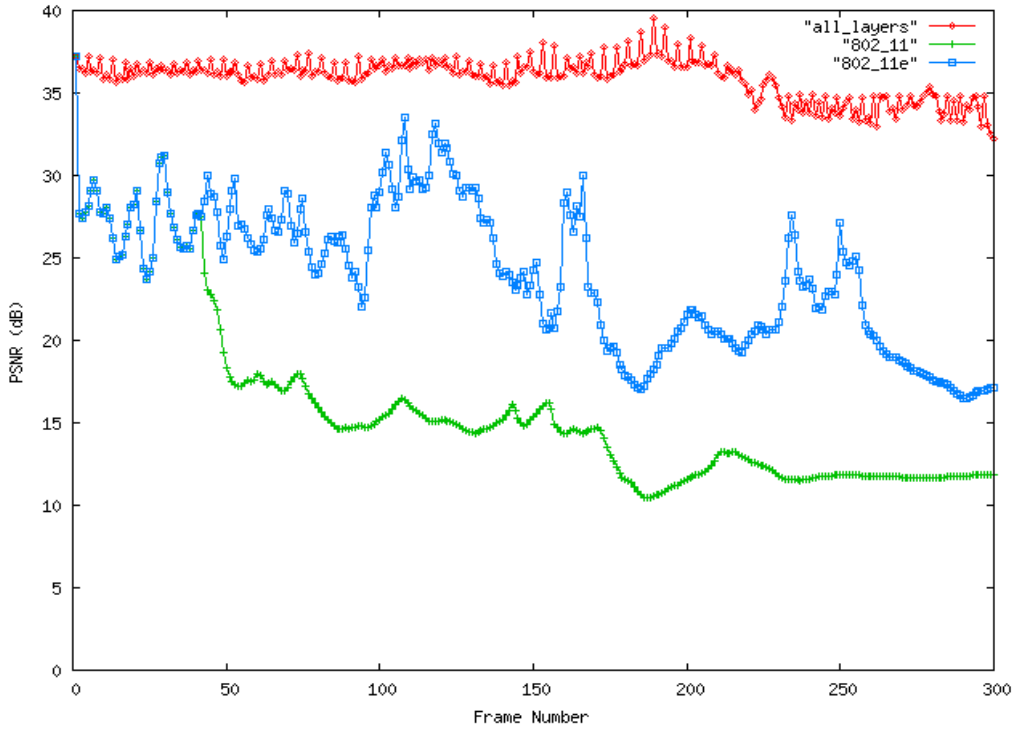

Fig. 9. PSNR values

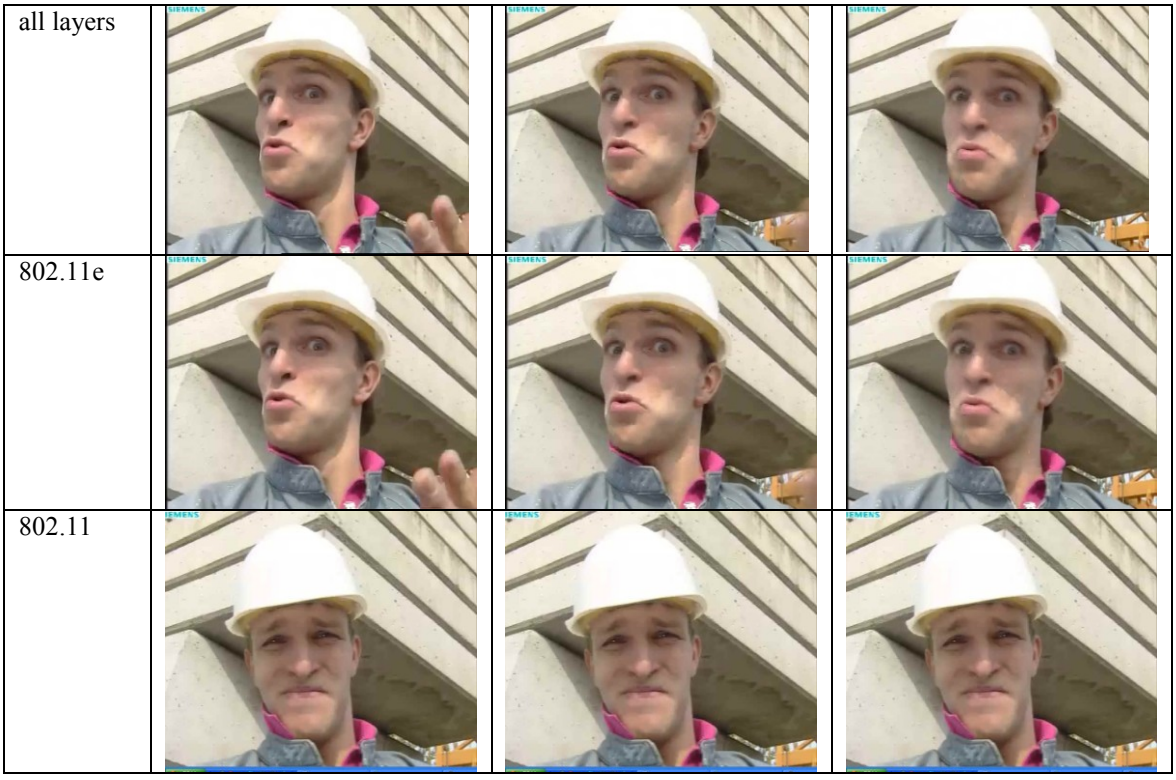

Fig. 10. Simulation-based visual comparisons

5. Conclusion

The objective of this paper is twofold. The first is to integrate SVEF and NS2 to create the myEvalSVC framework for the evaluation of H.264/SVC transmission in a simulated environment. Researchers who work on video coding can simulate the effects of a more realistic network on video sequences resulting from their coding schemes, while researchers who work on network technology can evaluate the impact of real video streams on the proposed network architecture or protocols.

The second is to provide a VirtualBox guest operating system image and a well-explained and illustrated website [21] that helps beginners easily repeat the examples of H.264/SVC transmission over IEEE 802.11 and 802.11e networks. They can start the evaluation from encoding the raw YUV video, parse the video content, prepare the NS2 traffic trace file, and perform the simulation. After the simulation, the network-level performance metrics such as packet loss rate and end-to-end delay can be obtained with the aid of programs provided in myEvalSVC. Moreover, the received video can be constructed through the process of filtering out very late and undecodable NALUs and through frame concealment. Lastly, the end-to-end application level metric, PSNR, can be calculated by comparison of the received final YUV video with the original raw YUV video. In addition, visual evaluation is also possible with the help of the YUVviewer program.

References

[1] "ITU-T recommendation H.264: Advanced video coding for generic audiovisual services," *International Telecommunication Union*, Nov. 2007.

[2] B. An, Y. Kim and O. J. Kwon, "Low-complexity motion estimation for H.264/AVC through perceptual video coding," *KSII Transactions on Internet and Information Systems*, vol. 5, no. 8, pp. 1444-1456, Aug. 2011. in *Proc. of* Article (CrossRef Link)

[3] H. J. Cho, D. Y. Noh, S. H. Jang, J. C. Kwon and S. J. Oh, "A new video bit rate estimation scheme using a Model for IPTV services," *KSII Transactions on Internet and Information Systems*, vol. 5, no. 10, pp. 1814-1829, Oct. 2011. Article (CrossRef Link)

[4] L. Zhou, H. H. Chen, "On distributed multimedia scheduling with constrained control channels," *IEEE Transactions on Multimedia*, vol. 13, no. 5, pp. 1040-1051, Oct. 2011. Article (CrossRef Link)

[5] L. Zhou, X. Wang, W. Tu, G. Muntean and B.Geller, "Distributed scheduling scheme over video streaming over multi-channel multi-radio multi-hop wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 28, no. 3, pp. 409-419, Apr. 2010. Article (CrossRef Link)

[6] H. Schwarz, D. Marpe and T. Wiegand, "Overview of the scalable video coding extension of the H.264/AVC standard," *IEEE Transaction on Circuits and Systems for Video Technology*, vol. 17, no. 9, pp. 1103-1120, Sep. 2007. Article (CrossRef Link)

[7] G. Bianchi, A. Detti, P. Loreti, C. Pisa, F. S. Proto, W. Kellerer, S. Thakolsri and J. Widmer, "Application-aware H.264 scalable video coding delivery over wireless LAN: Experimental assessment," in *Proc. of The Second International Workshop on Cross layer Design*, 2009. Article (CrossRef Link)

[8] A. Detti, P. Loreti, N. Blefari-Melazzi and F. Fedi, "Streaming H.264 scalable video over data distribution service in a wireless environment," in *Proc. of IEEE International Symposium on World of Wireless Mobile and Multimedia Networks*, Jun. 2010. Article (CrossRef Link)

[9] H. L. Chen, P. C. Lee and S. H. Hu, "Improving scalable video transmission over IEEE 802.11e through a cross-layer architecture," in *Proc. of The Fourth International Conference on Wireless and Mobile Communications*, Aug. 2008. Article (CrossRef Link)

[10] E. H. Putra, E. Supriyanto, J. Din and H. Satria, "Cross layer design of wireless LAN for telemedicine application," in *Proc. of Third Asia International Conference on Modelling &*

Simulation, 2009. Article (CrossRef Link)
[11] M. Li, Z. Chen, Y. P. Tan, "Cross-layer optimization for SVC video delivery over the IEEE 802.11e wireless networks," *Journal of Visual Communication and Image representation*, vol. 22, no. 3, pp. 284-296, Apr. 2011. Article (CrossRef Link)
[12] L. Zhou, H. C. Chao and A. Vasilakos, "Joint forensics-scheduling strategy for delay-sensitive multimedia applications over heterogeneous networks," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 7, pp. 1358-1367, Aug. 2011. Article (CrossRef Link)
[13] T. A. Le, H. Nguyen and H. Zhang, "EvalSVC - An evaluation platform for scalable video coding transmission," in Proc. of *IEEE 14th International Symposium on Consumer Electronics*, 2010 Article (CrossRef Link)
[14] A. Detti, G. Bianchi, C. Pisa, F. S. Proto, P. Loreti, W. Kellerer, S. Thakolsri and J. Widmer, "SVEF: An open-source experimental evaluation framework for H.264 scalable video streaming," *in Proc. of IEEE Symposium on Computers and Communications*, 2009. Article (CrossRef Link)
[15] SVEF-reference software, http://svef.netgroup.uniroma2.it/
[16] The Network Simulator-NS2, http://www.isi.edu/nsnam/ns/
[17] Microsoft Visual C++ 2008 Expression Edition, http://www.microsoft.com/express/Downloads/#2008-Visual-CPP
[18] Python Programming Language,http://www.python.org/
[19] Cygwin, http://www.cygwin.com/
[20] VirtualBox, http://www.virtualbox.org/
[21] myEvalSVC-reference software, http://hpds.ee.ncku.edu.tw/~smallko/ns2/svc.htm
[22] How to multicast H.264/SVC video over wired networks, http://hpds.ee.ncku.edu.tw/~smallko/ns2/svc_multicast_wired.htm
[23] V. K. Goyal, "Multiple description coding: Compression meets the network?," *IEEE Signal Processing Magazine*, vol. 18, no. 5, pp. 74-94, Sep. 2001. Article (CrossRef Link)
[24] Y. Wang and M. Claypool, "RealTracer-tools for measuring the performance of real video on the Internet," *Kluwer Multimedia Tools and Applications*, vol. 27, no. 3, Dec. 2005.
[25] 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, Mar. 2008. Article (CrossRef Link)
[26] How to evaluate MPEG video transmission using NS2 simulator, http://hpds.ee.ncku.edu.tw/~smallko/ns2/Evalvid_in_NS2.htm
[27] J. Klaue, B. Rathke and A. Wolisz, "Evalvid-A framework for video transmission and quality Evaluation," *The 13th International Conference on Modelling Techniques and Tools for Computer Performance Evaluation*, 2003. Article (CrossRef Link)
[28] Video Trace Library, http://trace.eas.asu.edu/
[29] A. Matrawy, I. Lambadaris and C. Huang, "MPEG4 traffic modeling using the transform expand sample methodology," *The Fourth IEEE International Workshop on Network Appliances*, 2002. Article (CrossRef Link)
[30] JSVM Software Manual, http://evalsvc.googlecode.com/files/SoftwareManual.doc
[31] IEEE Standard 802.11-1999, "Wireless LAN medium access control (MAC) and physical layer (PHY) specifications".
[32] IEEE Standard 802.11e-2005, "Wireless LAN medium access control (MAC) and physical layer (PHY) specifications, Amendment 8: medium access control (MAC) quality of service enhancements".
[33] YUVviewer, http://wftp3.itu.int/av-arch/jvt-site/software_tools/

Chih-Heng Ke received his B.S. and Ph.D degrees in electrical engineering from National Cheng-Kung University, in 1999 and 2007. He now is an assistant professor of Computer Science and Information Engineering, National Quemoy University, Kinmen, Taiwan. His current research interests include multimedia communications, wireless networks, and QoS networks.