Application-aware H.264 Scalable Video Coding delivery over Wireless LAN: experimental assessment

Giuseppe Bianchi, Andrea Detti, Pierpaolo Loreti,
Claudio Pisa, Francesco Saverio Proto
University of Roma Tor Vergata, Italy
{name.lastname}@uniroma2.it

Wolfgang Kellerer, Srisakul Thakolsri, Joerg Widmer
DOCOMO Euro-Labs
Munich, Germany
{lastname}@docomolab-euro.com

***Abstract*—This paper is among the first works to document experimental results for application-aware H.264 Scalable Video Coding (SVC) support over Wireless LANs. Application-aware support is achieved by introducing a bandwidth throttling device, called Virtual BottleNeck (VBN), before the WLAN Access Point. Throttling is set to a bandwidth slightly smaller than the actual WLAN capacity (either known or estimated), so that all packet/frame losses occur inside the VBN. Here, loss events are controlled by a scheduling mechanism devised to operate with information taken from the H.264 Network Abstraction Layer Units (NALUs). Despite its relative simplicity, the implemented scheduler exhibits effective video adaptation performance and close to optimal bandwidth efficiency. Setting up the trial was not trivial due to the lack of suitable publicly available tools. We filled this gap by implementing and integrating several separate software modules, e.g., streaming server, NALU dependency filtering, video frame concealment, etc. As a final result, the experimental trial supports the full delivery chain for an SVC stream with the only limitation of an off-line stream conversion for play-out and Peak Signal-to-Noise Ratio (PSNR) measurement purposes, due to the unavailability of real time SVC players.**

I. Introduction

Scalable Video Coding (SVC) is an effective solution for video streaming over a channel with time-varying bandwidth and it is a particularly promising candidate [1], [2], [3], [4] for video delivery over 802.11 (Wi-Fi) [5] Wireless Local Area Networks (WLANs). Similar to other wireless networks, wireless channel impairments affect the WLAN physical transmission rate assigned to the user through rate adaptation mechanisms [6], [7]. The actual throughput achieved by a specific user not only depends on the number of users sharing the same wireless channel, but also on the dynamic changes of the channel quality experienced by this user as well as those experienced by *all* other users. This is caused by a subtle side-effect emerging from the WLAN Medium Access Control operation known as Performance Anomaly [8].

While SVC concepts have been known for about 20 years, only recently (2007) an SVC standard was finalized in the framework of the ITU H.264 advanced video coding standards [9]. The standardization group provides the reference software, JSVM [10], for encoding and decoding purposes. However, despite the extensive research work carried out (e.g. in the frame of the European project Astrals [11]), no public domain software appears currently made available for the many other phases emerging in the end-to-end video delivery chain, including streaming of the video over IP packets through the Real-time Transport Protocol (RTP) and the reconstruction of H.264 SVC received videos in the presence of random frame losses (the current version v9.15 of JSVM not being able to cope with arbitrary losses).

Most of the experimental work reported in the literature focuses on SVC adaptation through application layer end-to-end signalling mechanisms. For instance, [12] develops an RTP/RTCP proxy which uses RTCP feedback for SVC adaptation purposes. Another proxy prototype has been presented in [13], where MPEG-21 Digital Item Adaptation is used. An SVC server is developed in [14] as part of an optimized MPEG-21 framework for an heterogeneous network scenario.

Rather, to the best of our knowledge, our is among the first works which experimentally demonstrate the viability of SVC in-network [15], [16] adaptation through an application-aware traffic scheduler (as opposed to application layer proxy operation) devised to adaptively drop H.264 SVC video frames, (more precisely, Network Abstraction Layer Units). The ability to selectively drop frames is accomplished through the introduction of a bandwidth throttling device, called Virtual BottleNeck (VBN), placed outside the WLAN Access Point (AP). It reduces the link bandwidth to a value slightly smaller than the actual WLAN capacity. Since all packet loss due to traffic overload is transferred from the WLAN network to the VBN, application-aware scheduling mechanisms can be easily implemented, without requiring modification in the (legacy) APs. As a practically important side result of our work, we provide all the software components we had to develop for running experimental campaigns in the form of free and open source code[1].

II. Virtual BottleNeck

The approach proposed for application-aware support of H.264 SVC delivery over WLANs is illustrated in Figure 1. In this paper we restrict our investigation to the case of "downlink video streaming". This is representative of a video-on-demand

[1]The source code of the software as well as all the videos produced and discussed throughout the remainder of this paper can be found at the URL `http://netgroup.uniroma2.it/iwcld09/` - for download purposes note that videos are 300-400 MB each.

978-1-4244-3302-5/09/$25.00 ©2009 IEEE

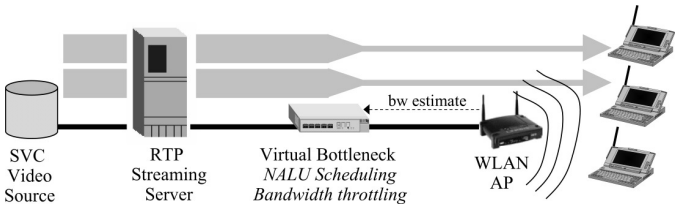

Fig. 1. Network scenario with video server, VBN, WLAN AP, etc.

scenario, where end users connected to a WLAN hot-spot independently access one or more video servers placed in the wired network.

The idea behind the Virtual BottleNeck (VBN) illustrated in Figure 1 is very simple, but practical and effective. It emerges from the observation that MAC-layer frame losses only rarely occur in a Wireless LAN because of channel quality impairments. In fact, starting from Auto Rate Fallback [6], several rate adaptation mechanisms [7] have been proposed to improve frame delivery, by estimating the channel quality and/or measuring the experienced frame loss ratio, and then switching to a suitable modulation scheme. The 802.11 MAC function retransmits MAC frames corrupted because of channel errors or channel access collisions. As a result, a MAC frame is lost completely only if it reaches a maximum number of retransmissions. In the 802.11 standard, this is a relatively large value (the default settings being 4 – Short Retry Limit – and 7 – Long Retry Limit – retransmissions, depending on the length of the MAC frame [5]). Therefore, in normal conditions, the MAC frame loss ratio seen by higher layers is typically low. It only becomes significant if severe channel degradation occurs, so harshly that even rate adaptation to the minimal available transmission rate is not sufficient.

We can thus assume that the majority of all MAC frame losses occur at the AP buffer. Loss events clearly occur when the load offered to the AP is greater than the maximum throughput available at the AP. In general, the time-varying capacity $C_{AP}(t)$ depends on i) the number of stations competing with the AP for channel access and ii) the individual transmission rates of *all* competing stations [8].

The Virtual BottleNeck is a traffic control box placed in the wired network before the AP. It intercepts all the traffic offered to the AP itself. Its goal is to enforce a traffic throttling function devised to prevent the traffic offered to the AP from overflowing the available capacity $C_{AP}(t)$. Provided that the throttling function is able to follow the variations in time of the AP capacity, and provided that a sufficient AP buffering capability is available and a sufficient bandwidth margin is deployed between the traffic offered by the VBN and the actual AP capacity, the ultimate result is that the AP buffer will never saturate, and hence no frame loss will occur at the AP itself. Rather, all the losses will occur inside the VBN box.

Several mechanisms exist for the run-time estimation of the available AP capacity and the consequent dynamic control of the throttling function, e.g., [17], [18], [19], [20]. However, the details of this estimation are outside the scope of the present paper. Here, we are interested in taking full advantage of the VBN in exploiting application layer information for scheduling traffic.

We remark that since the VBN is a separate control entity, it can easily be deployed in any pre-existing WLAN infrastructure with legacy Access Points. If the WLAN supports 802.11e Quality of Service enhancements (as is the case in our experimental set-up), these can be exploited by configuring the VBN to set the IP Type Of Service (TOS) field to the value 160 (for WMM - Wireless Multimedia - compliant APs) so that MAC frame transmission occurs with EDCA *video access category*.

III. Application-aware VBN scheduling for H.264 SVC traffic

The proposed application-aware scheduling algorithm operates at the network layer. Its service policy is based on the control information contained in the header of the H.264 SVC Network Abstraction Layer Units (NALUs). We first give an overview of this control information, and then discuss its usage by the proposed scheduler.

A. H.264 SVC background

An H.264 SVC stream is a sequence of NALUs. A NALU is formed by a header and a payload carrying the actual encoded video frame. The NALU header contains information about the NALU type and its relevance in the decoding process. From the information reported in the NALU header (see full details in [9], or [13]), we are specifically interested in the three parameters called dependency_id (DID), temporal_id (TID), and quality_id (QID). Each parameter determines a specific scalability facility. DID allows *Coarse Grain Scalability*, TID allows *Temporal Scalability* and QID allows *Medium Grain Scalability*.

Coarse Grain Scalability (CGS) provides the ability to *coarsely* adapt a video performance; e.g., video's spatial resolution from CIF to 4CIF. The video should be encoded with a suitable set of coarse enhancement sub-streams, called *dependency-layers*. DID is the identification of the dependency-layer of the NALU. The decoding of a NALU belonging to the dependency-layer $did > 0$ depends on NALUs of dependency-layer $did - 1$, with the same value of TID and QID. Following this dependency rule, we can coarsely reduce video quality by removing NALUs with a DID greater than a specific value. For simplicity, we do not consider Coarse Grain Scalability in the rest of this paper. However, extending our work to CGS is straightforward.

Temporal Scalability provides the ability to adapt the video frame-rate. The TID specifies the *temporal-layer* of the NALU, i.e., the "frame-rate sub-stream". A NALU belonging to the temporal-layer $tid > 0$ and with $qid = 0$ depends on NALUs of temporal-layer $tid - 1$, with the same DID and QID. Following this rule, a frame-rate scaling should be accomplished by removing NALUs with a TID greater than a specific value.

Medium Grain Scalability (also called *progressive refinement*) allows the adaptation of video quality (i.e., PSNR). The

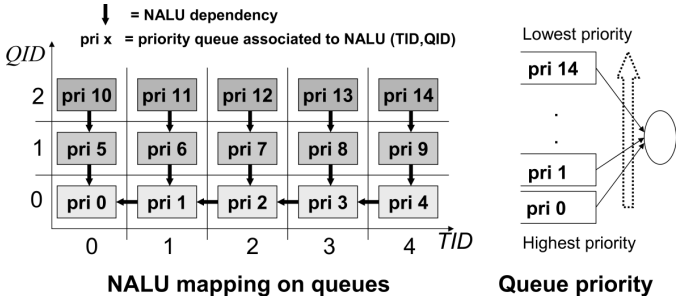

Fig. 2. NALU scheduler

video should be appropriately encoded with a set of quality enhancement sub-streams, called *quality-layers*. A quality-layer reduces the encoding quantization error, and thus improves the PSNR. The QID identifies the quality-layer of the NALU. A NALU belonging to the quality-layer $qid > 0$ depends on NALUs of quality-layer $qid - 1$, with the same DID and TID. Following this dependency rule, the quality scaling should be accomplished by removing NALUs with a QID greater than a specific value.

Overall, with reference to temporal and medium grain scalability, the dependency rules can be summarized as follow, where the arrow means "depends on"

$$\begin{array}{llll} (tid > 0, & qid = 0) & \rightarrow & (tid - 1, \quad qid = 0) \\ (tid \geq 0, & qid > 0) & \rightarrow & (tid, \qquad qid - 1) \end{array}$$

B. H.264 SVC application-aware scheduler

The design target of our proposed application-aware scheduler is to exploit H.264 SVC NALU types and their dependencies to

1) accomplish an efficient usage of the wireless resource by avoiding to transfer NALUs that will not be decoded by the receiver because of missing dependencies;
2) provide a smooth adaptation of the video quality versus changes in the available capacity $C_{AP}(t)$ or the offered load of the video traffic.

These two goals can be accomplished through a priority queuing discipline, dedicating a *separate* queue to each possible TID-QID combination. Considering that the default range for TID values is from 0 to 4, and considering two additional enhancement quality-layers (i.e., QID values in the range from 0 to 2), we deploy $5 \times 3 = 15$ limited-size queues, with queue #0 having the highest priority and queue #14 having the lowest one, as shown in Figure 2. An incoming NALU is delivered to a queue #n according to the following classification rule:

$$n = 5qid + tid$$

This ensures that a NALU x will have a lower service priority than the NALUs it depends upon (first goal). Anyway, it may exceptionally happen that at the receiver side, a delivered NALU lacks other NALUs it depends on, since the dropping decision is taken locally at each queue. For instance, a NALU with a given tid and $qid = x$ may be dropped from its queue due to a peak load fluctuation, while the lower priority queue associated to the same tid but $qid = x+1$ may not experience the same fluctuation.

Finally, we observe that in the presence of congestion, the NALUs of the higher quality layers will be discarded first, and only later the NALUs of the base layer (second goal).

IV. Implementation Issues and Experimental Setup

To cope with the unavailability of essential software components, we developed an hybrid online/offline testbed. While the NALUs are delivered in real-time, several pre-processing and post-processing mechanisms can only be applied off-line. The result is a testbed, which is functionally equivalent to a purely online video delivery process. The H.264 delivery chain deployed in our testbed can be summarized in the following steps.

(1) We first encode a raw YUV video file into SVC using the *JSVM H264Encoder*.

(2) From the resulting H.264 encoded video file we generate a packet trace file with the *JSVM BitStreamExtractor* tool. This file is used as an hint track by the video streamer (described below) for timing and packetization purposes.

(3) We feed the H.264 video and the packet trace to a custom *video streamer* module, which constructs a simplified RTP header for each NALU. We map one NALU to each RTP unit and send RTP packets according to the video frame-rate. Large NALUs are thus split through IP layer fragmentation.[2]

(4) IP fragments received at the VBN are reassembled using the standard IP_conntrack Linux Kernel facility. Then entire IP packets (i.e., NALUs) are sent to an internal virtual interface (an Intermediate Queuing Module of Linux) where we implement the application-aware scheduling using the IPROUTE 2 tools. At the exit of this virtual-interface, IP packets are again fragmented and transferred to the output interface.

(5) IP fragments are received by the WLAN AP, which transmits them to the end devices. At the receiver side, all the received NALUs are collected in an H.264 JSVM compatible trace file. We post-process the trace with a custom software module (*NALU-Filter*) devised to i) discard NALUs received after a pre-set maximum playout delay (5 seconds in our experiments) and ii) check NALU dependencies and discard the NALUs for which dependencies are missing.[3] Besides the filtered H.264 trace file, the NALU-Filter also returns the number of received NALUs and the number of filtered NALUs. These are two basic metrics to evaluate if wireless resources are used efficiently. If no NALUs are filtered, no resources are wasted.

[2]The alternative would have been to use RTP fragmentation. This allows to deal with NALU sizes greater than 64 kbytes (the upper limit for UDP datagrams), but would require to i) implement RTP fragmentation/reassembly from scratch, and ii) to reassemble NALUs at the VBN before their scheduling. Unlike AVC, the RTP header field does not contain SVC information and the tuple (DID,TID,QID) is carried as RTP payload. Hence, it is available only on the first RTP fragment. In our experiments, the 64 kbytes upper limit for the NALU size was never reached with SVC, while for AVC we solved this issue by dividing each video frame in two slices.

[3]This latter check is strictly necessary since the JSVM H264Decoder (v9.15) hangs if a NALU dependency fails.

TABLE I
VIDEO TEST-SEQUENCE PARAMETERS

	BL (kbps)	MG1 (kbps)	MG2 (kbps)	Full (kbps)	PSNR (dB)
SVC(A)	648	907.3	1304.7	2860	36.64
SVC(B)	1295	815	637	2748	36.50
AVC	2693	-	-	2693	36.49

(6) The filtered H.264 trace file is used to reconstruct an H.264 video, which is in turn decoded with the JSVM H264Decoder, thus obtaining an uncompressed YUV file.

(7) This YUV file may contain missing frames, since relevant NALUs may have been lost. To maintain the original temporal sequence and to simplify PSNR measurements, we develop a simple *Frame-Filler* tool. It outputs a final YUV video that has the same number of frames as the original one. When a frame is missing, the Frame-Filler inserts the last available received frame instead, which is a very basic form of error concealment.

(8) Finally, the resulting YUV file and the original one are compared with the *JSVM PSNR* evaluation tool to assess overall video quality.

V. EXPERIMENTAL RESULTS

Results are provided for the following setup. We use a 10 second clip of a publicly available 4CIF YUV video (soccer game) at 30 fps. A 50 second video sequence is generated by concatenating 5 repetitions of the video. Through JSVM, we encode the 50 seconds video with three different approaches that are detailed in table I. The SVC (A) and SVC (B) are encoded with Medium Grain Scalability and have three quality-layers (base-layer BL and two enhancement layers MG1 and MG2). SVC (B) has a larger base layer than (A), and the enhancement layers reduce in bitrate as the QID increases. The opposite behavior is chosen for (A). Finally, the AVC (H.264 Advanced Video Coding) video only uses a base layer. All the encoded videos have approximately the same average bitrate, similar bitrate fluctuations (about $\pm$ 30%) and the same client-side playout buffer of 5 seconds. For the experiments, we assume that a first user starts retrieving the video stream at time 0. A new user arrives every 8 seconds (240 frames) and begins the downloading of the same video stream. Video performances are measured for the first user.

The experiments are based on an indoor WLAN deployment with 5 stations associated to an AP. All stations experience good average channel conditions (the distance to the AP is less than 2 meters in LOS conditions) and support the maximum 11 Mbps 802.11b physical layer rate with no losses. The VBN has been throttled to 6.0 Mbps, a value just below the measured MAC throughput at the AP of about 6.3 Mbps. This guarantees, as we confirmed in subsequent measurements, that no MAC frames are lost at the AP buffer. Moreover, we also perform tests with the WLAN physical layer rate reduced to 2 Mbps; in this cases the VBN is throttled to 1.5 Mbps. For the evaluation, results are reported in terms of a video quality metric (PSNR) as well as a delivery efficiency metric.

A. Impact of the VBN

Figure 3 shows the Y-PSNR (luminance) over time, measured for the video stream SVC (A) delivered to the first user with and without VBN scheduling, with respect to the original, pre-encoding raw video. The PSNR is compared to two reference curves: i) the ideal PSNR (top curve labeled "all layers") of the stream for the case of no NALU loss, where the resulting PSNR depends only on the degradation due to the encoding process, and ii) the PSNR provided by the base layer only (labeled "base layer"), assuming that all base layer NALUs are received and all NALUs of other layers are dropped.

Figure 3 confirms that *the delivery performance of H.264 SVC is poor without application-aware scheduling enforced by the VBN, i.e., when MAC frames, and as a consequence NALUs, are dropped randomly*. A sudden severe PSNR degradation occurs under overload conditions. The resulting video frequently "freezes" (meaning that several video frames were lost), and the overall video quality is unacceptable. With an average total video rate of 2.86 Mbps, this happens when three streams are delivered. The PSNR does not degrade further when additional streams are admitted. This is due to the fact that the PSNR given by the comparison of two random frames from the same test sequence is around 15 dB, as confirmed by further experiments (not shown here). Thus, this is the lowest PSNR value we can expect.

Conversely, the application-aware scheduler allows for a smooth degradation of the H.264 SVC stream. When all 5 users share the channel, they achieve an average rate of 700 kbps per user. The PSNR approaches that of the base layer alone, which is the expected behavior, given that the base layer uses on average 650 kbps.

B. H.264 SVC versus AVC

Figure 4 shows the performance advantages of SVC (A) compared to AVC. In both cases, frame losses are controlled through the VBN scheduling. Again, the results confirm that *SVC is a much more suitable coding mechanism in a scenario where large variations in the available capacity occur*. When three or more stations compete and an overload emerges, SVC reduces the quality of the video, which results in a significantly smoother PSNR degradation compared to AVC's temporal scalability adaptation.

While somewhat obvious, this consideration has important practical implications on how to encode H.264 SVC streams when they are delivered over a WLAN. Recalling that the H.264 SVC base layer is AVC encoded, we expect that under severe overload conditions where it is impossible to transmit the base layer without NALU losses, the quality degradation will become significant. This can be seen from Figure 5. The setup is the same as that of Figure 3, with the fundamental difference that the WLAN is configured to provide only a 2 Mbps PHY rate and the VBN rate is set to 1.5 Mbps. The key difference between Figure 5 and Figure 3 is that video stream scaling occurs immediately. The available bandwidth is lower than the total bandwidth requirement for all the layers

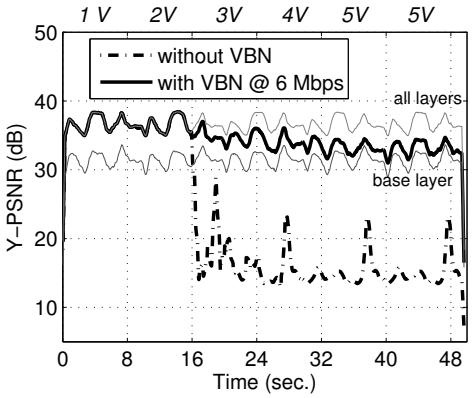

Fig. 3. SVC (A) with/without scheduler and WLAN @ 11 Mbps

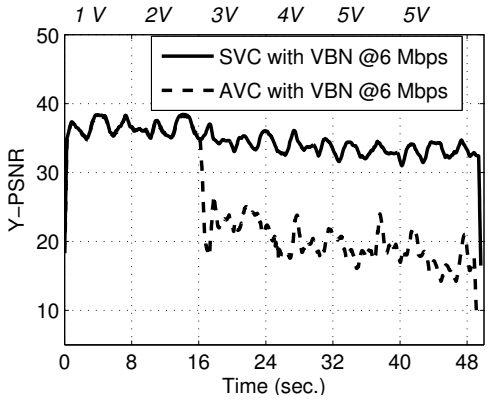

Fig. 4. SVC (A) versus AVC with scheduler and WLAN @ 11 Mbps

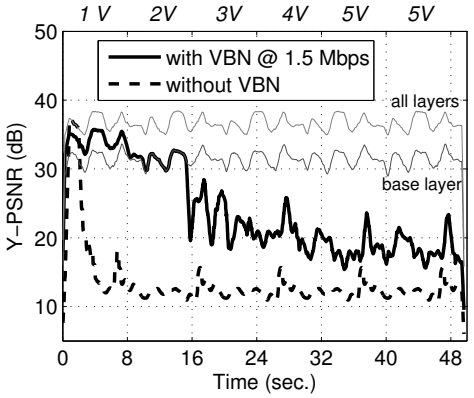

Fig. 5. SVC (A) with/without scheduler and WLAN @ 2 Mbps

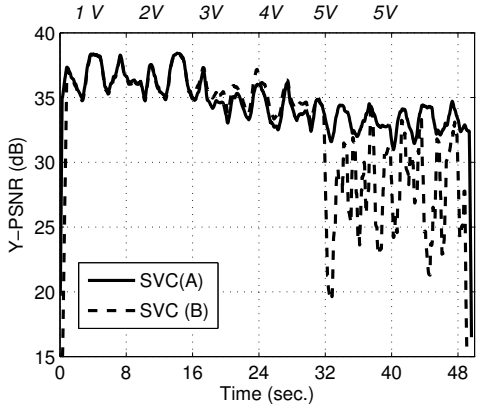

Fig. 6. SVC (A) and SVC (B) with scheduler and WLAN @ 11 Mbps

and, much more importantly, when three or more streams compete, adaptation is required also for the base layer. As a consequence, performance drops as in the AVC case (although not nearly as dramatically as in the case of no application-aware scheduling).

This insight is especially significant for the following reason. In Figure 6, we run the experiments for two different encoding choices: SVC (A) and SVC (B). The figure clearly highlights that it is preferable to reduce the size of the base-layer, especially if this produces only a marginal degradation of the overall video encoding efficiency (as in our experiments). While for SVC (A), degradation affects only the enhancement layer NALUs, SVC (B) experiences base-layer NALU losses and the resulting AVC temporal scalability reduces PSNR much more severely.

C. Scheduler impact and performance

In Table II, we provide some summarizing results on delivery efficiency for the previous experiments. In addition, the last row shows the performance experienced by an AVC stream for the scenario of Figure 5 with the VBN throttled to 1.5 Mbps.

The table reports three performance metrics: *transmission efficiency*, defined as the percentage of NALUs received by the client which can be used for decoding (i.e., for which encoding dependencies and playout delay conditions are satisfied), the total number of video frames that could not be decoded (out of the 1490 frames transmitted), and the average PSNR over the whole 50 seconds of the experiments.

The most interesting result provided in the table is the transmission efficiency of the considered scheduler. Though the scheduler does not guarantee that all NALU dependencies will ultimately be satisfied, the performance for both AVC and SVC cases with VBN is close to 100% efficiency. Without the scheduler, transfer efficiency is always very poor.

The column reporting the number of missing video frames gives an impression of the perceived quality of the final video stream (again, refer to the web site at `http://netgroup.uniroma2.it/iwcld09` for visual comparison of the actual streams). In the 6 Mbps scenario, SVC yields 0 missing frames, compared to the 845 misses of the AVC. In the low rate 1.5 Mbps scenario, almost all frames are missing for AVC and SVC without scheduler, while a reasonable quality is achieved in the SVC case with scheduler. Looking at the actual video stream, we see that frames are missing periodically and the SVC has scaled to operate at about the half of the frame rate.

VI. Conclusions and Further Research Directions

In this paper we have presented experimental results dealing with application-aware H.264 Scalable Video Coding (SVC) delivery over Wireless LANs. To the best of our knowledge,

TABLE II
SUMMARY PERFORMANCE

Video Type	Scenario (VBN, WLAN rate)	TX Efficiency	# Missing Frames	Average PSNR
SVC (A)	6Mbps, 11Mbps	100.00 %	0	34.67
SVC (A)	no VBN, 11Mbps	54.64 %	941	22.52
AVC	6Mbps, 11Mbps	99.92 %	845	25.25
SVC (B)	6Mbps, 11Mbps	98.39 %	121	32.75
SVC (A)	1.5Mbps, 2Mbps	100.00 %	799	24.02
SVC (A)	no VBN, 2Mbps	21.33 %	1424	13.51
AVC	1.5Mbps, 2Mbps (not shown in the fig.)	99.39 %	1391	16.37

these results are among the first that are based on experimentation involving H.264 SVC in a real wireless testbed. To accomplish this goal, we had to develop several software components to provide functions such as streaming, NALU dependency filtering, concealment of missing frames, etc., which are not readily supported by off-the-shelf publicly available H.264 SVC-related software.

Our results prove the viability of the Virtual BottleNeck (VBN) mechanism and the application-aware scheduling. The VBN performs bandwidth throttling before the traffic is delivered to the WLAN Access Point, so that all packet loss occurs inside the VBN. The scheduler prioritizes the packets according to their importance for the video quality and also takes the dependency of the packets into account. This allows to discard packets at the VBN in a manner that has the least negative impact on the overall video quality. The scheduler maintains separate queues for the different priority levels.

Two major conclusions can be drawn from our preliminary results. First, significant performance improvements can be achieved even with very simple scheduling approaches (we developed an approach "just" based on priority queuing). Second, results show that SVC should be encoded with a base layer that is as small as it can be without significantly affecting overall encoding efficiency. This allows to cope well with the specific bandwidth characteristics and fluctuations expected in a WLAN. We believe that such a WLAN-aware SVC encoding should be further explored to better understand to what extent a base layer reduction is possible without impairing SVC encoding efficiency, and to understand the impact of different encoding choices for the remaining enhancement layers.

The experimental testbed highlighted in this paper can be improved in several ways, which are currently ongoing work. This includes i) extending the approach from application-aware to full cross-layer, by further exploiting wireless channel per-ssession rate adaptation and queueing status information; this may require to abandon the VBN approach and directly integrate the scheduler inside the WLAN AP, ii) identification of more efficient (quality-aware) scheduling mechanisms, and iii) extending the proposed approach to cope with more complex hybrid scenarios where both uplink streaming and downlink streaming may coexist.

VII. ACKNOWLEDGEMENTS

The authors wish to acknowledge Andrea Magurano for the support in producing video coded traces and running experimental campaigns. Claudio Pisa has been partially supported by the Italian project PRIN-SESAME.

REFERENCES

[1] M. van der Schaar, S. Krishnamachari, S. Choi, X. Xu, "Adaptive cross-layer protection strategies for robust scalable video transmission over 802.11 WLANs", IEEE J. on Selected Areas in Communications, Vol. 21, No. 10, Dec. 2003, pp. 1752-1763

[2] H. Liqiao, D. Raychaudhuri, Liu Hang, K. Ramaswamy, "Cross layer optimization for scalable video multicast over 802.11 WLANs", 3rd IEEE Consumer Communications and Networking Conference, Jan. 2006

[3] Y. P. Fallah, P. Nasiopoulos, H. Alnuweiri, "Efficient Transmission of H.264 Video over Multirate IEEE 802.11e WLANs", EURASIP Journal on Wireless Communications and Networking, 2008

[4] Chuan Heng Foh, Yu Zhang, Zefeng Ni, Jianfei Cai, King Ngi Ngan, "Optimized Cross-Layer Design for Scalable Video Transmission Over the IEEE 802.11e Networks", IEEE Trans. on Circuits and Systems for Video Technology, Vol. 17, No. 12, Dec. 2007

[5] IEEE Standard 802.11-2007, IEEE Standard for Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks-Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications; June 2007.

[6] A. Kamerman, L. Monteban, "WaveLAN-II: A high performance wireless LAN for the unlicensed band", Bell Labs Technical Journal, 1997, volume 2, issue 3, pp. 118-133.

[7] K. Ramachandran, H. Kremo, M. Gruteser, P. Spasojevic, I. Seskar, "Scalability Analysis of Rate Adaptation Techniques in Congested IEEE 802.11 Networks: An ORBIT Testbed Comparative Study", IEEE WoWMoM 2007.

[8] M. Heusse, F. Rousseau, G. Berger-Sabbatel, A. Duda, "Performance anomaly of 802.11b", IEEE Infocom, 2003.

[9] Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VQEG, Joint Scalable Video Model. Doc. JVT-X202, July 2007.

[10] Joint Scalable Video Model - reference software: http://ip.hhi.de/imagecom_G1/savce/downloads/SVC-Reference-Software.htm

[11] T. Zahariadis, "ASTRALS Project presentation", IBC 2007, Amsterdam, September 2007

[12] I. Kofler,M. Prangl, R. Kuschnig, H. Hellwagner, "An H.264/SVC-based adaptation proxy on a WiFi router", 18th ACM Int. Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV), Braunschweig, Germany, May 2008, pp. 63-68.

[13] R. Kuschnig, I. Kofler, M. Ransburg, H. Hellwagner, "Design options and comparison of in-network H.264/SVC adaptation", J. of Visual Commun. and Image Representation, Vol. 19, No. 8, Dec. 2008, pp. 529-542.

[14] M. Eberhard, L. Celetto, C. Timmerer, E. Quacchio, H. Hellwagner, F.S. Rovati, "An interoperable streaming framework for Scalable Video Coding based on MPEG-21", 5th Int. Conf. on Visual Information Engineering, Aug. 2008. VIE 2008, pp.723-728

[15] D. Wu, Y. T. Hou, Y. Q. Zhang, "Scalable video coding and transport over broadband wireless networks", Proc. IEEE, vol. 89, no. 1, pp. 6-20, 2001.

[16] G. Bianchi, A. T. Campbell, R. Liao, "On utility-fair adaptive services in wireless networks", 6th Int. Workshop Quality of Service (IWQOS'98), Napa Valley, CA, May 1998

[17] H. K. Lee, V. Hall, K. H. Yum, K. Kim, E. Kim, "Bandwidth estimation in wireless LANs for multimedia streaming services", IEEE Int. Conf. on Multimedia and Expo (ICME), 2006.

[18] C. Sarr, C. Chaudet, G. Chelius, I. Lassous, "A node-based available bandwidth evaluation in IEEE 802.11 ad hoc networks", 11th Int. Conf. on Parallel and Distributed Systems (ICPADS), 2005.

[19] S. Shah, K. Chen, K. Nahrstedt, "Available bandwidth estimation in ieee 802.11-based wireless networks", ISMA CAIDA Workshop on Bandwidth Estimation (BEst), 2003.

[20] M. Neilsen, K. Ovsthus, L. Landmark, "Field trials of two 802.11 residual bandwidth estimation methods", IEEE Int. Conf. on Mobile Adhoc and Sensor Systems (MASS), 2006.