Cross-layer H.264 Scalable Video Downstream Delivery Over WLANs

Giuseppe Bianchi, Andrea Detti,
Pierpaolo Loreti, Claudio Pisa
University of Tor Vergata, Rome, Italy

Srisakul Thakolsri, Wolfgang Kellerer, Joerg Widmer
DoCoMo Euro-Labs, Munich, Germany

***Abstract*—Thanks to its in-network drop-based adaptation capabilities, H.264 Scalable Video Coding is perceived as an effective approach for delivering video over networks characterized by sudden large bandwidth fluctuations, such as Wireless LANs. Performance may be boosted by the adoption of application-aware/cross-layer schedulers devised to intelligently drop video data units (NALUs), so that i) decoding dependencies are preserved, and ii) the quality perceived by the end users is maximized. In this paper, we provide a theoretical formulation of a QoE utility-optimal cross-layer scheduling problem for H.264 SVC downlink delivery over WLANs. We show that, because of the unique characteristics of the WLAN MAC operation, this problem significantly differs from related approaches proposed for scheduled wireless technologies, especially when the WLAN carries background traffic in the uplink direction. From these theoretical insights, we derive, design, implement and experimentally assess a simple practical scheduling algorithm, whose performance is very close to the optimal solution.**

I. Introduction

Wireless Local Area Networks [1] are characterized by capacity constraints which vary over time. Arrival and departure of traffic sessions competing for access to the shared medium may cause huge fluctuations in the capacity available to the access point or to stations. Channel condition changes may trigger physical (PHY) rate adaptation mechanisms [2], [3] which yield frequent and abrupt step-wise changes of the available rate. Furthermore, because of the well known "performance anomaly" of the WLAN Medium Access Control operation [4], the actual throughput achieved by a station experiencing a good channel quality is severely affected by the PHY rate changes due to variations in the channel quality experienced by *other* stations.

Scalable Video Coding (SVC) is a very promising coding technique that allows to adapt to such challenging network conditions [5]. Its basic concepts have been investigated by the research community for almost two decades, and the recent finalization of an SVC specification in the framework of the ITU H.264 advanced video coding standards family [6] in 2007 is likely to further stimulate its deployment. An SVC stream is composed of multiple "layers" which carry incremental video enhancement information. As such, it can be adapted to a capacity throttling or fluctuation simply by dropping (in part or in full) one or more enhancement layers, thus obviating the need for costly transcoding operations or the provision of multiple streams at different quality levels. H.264 SVC provides a very smooth adaptation process at a level of granularity down to individual dropping decisions for single video layer protocol data units (also called Network Abstraction Layer Units or NALUs in H.264 notation).

Adaptation of the video stream to the available network capacity should be done in an *application-aware* fashion. Dropping NALUs of a lower layer may make corresponding higher layer NALUs useless because of missing decoding dependencies. Many SVC adaptation solutions have been proposed [7], [8], [9], [10], [11]. These are either implemented at the remote video server, or deployed within the network (e.g. at middleboxes such as proxies or at wireless base stations/access points). The adaptation mechanism requires feedback about the available capacity, which can be provided with lower overhead and in a more timely manner, the closer to the capacity bottleneck the adaptation occurs. For example, it is possible to leverage locally available detailed channel state information for adaptation at base stations or access points.

A. Problem Statement

An issue largely addressed in literature concerns the allocation of wireless channel resources so that the video quality is optimized with respect to some chosen performance metric. For adaptive video streams, the common approach is to rely on (Quality of Experience, QoE) utility curves, expressing the user-perceived quality which model the application utility (e.g. expressed in terms of Mean Opinion Score (MOS), or Peak Signal to Noise Ratio (PSNR)) versus the network resources committed to that stream. When considering a fixed capacity network, the resulting optimization problem becomes straightforward. It suffices to distribute a known and constant pool of resources, namely the overall capacity, to the different streams so that the resulting overall utility is, for instance, maximized. Many papers have generalized this problem to the multi-rate case, most notably in the context of scheduled technologies such as 802.16, 3G/LTE, etc. In this case, there is no notion of "total" wireless network capacity, as the PHY rate of the wireless terminals depends on the channel conditions and the specific modulation and coding techniques employed. However, the optimization problem can be reduced to that of allocating a known and constant pool of resources, by considering the available channel time, or equivalently the PHY symbol rate, instead of capacity.

At a first glance, the WLAN hot-spot scenario comprising downlink video streaming appears very similar. In fact, the

problem of allocating resources to the different video streams becomes a centralized scheduling problem, as the AP may take appropriate decisions on how to allocate its transmission capacity to the connected users. A closer look reveals, however, that the case differs substantially from the previous ones, especially when the AP is not the only active station in the network. When two or more stations are accessing the network, channel resources are managed in terms of transmission opportunities, rather than in terms of capacity or channel time. Thus the utility optimization problem cannot be expressed as the sharing of a given amount of capacity. Instead, it is necessary to take into account that, in a multi-rate scenario, the channel time dedicated to the video streams transmitted by the AP depends upon the scheduling decisions taken by the AP itself. Although the consequences of the MAC layer's sharing of transmission opportunities rather than time is well known and understood, to the best of our knowledge no prior work appears to relate these insights to the utility optimization problem discussed above.

B. Contributions of the Paper

The paper addresses the problem of scheduling downlink video traffic in Wireless LAN hotspots in order to maximize the utility provided to the customers in terms of QoE. The paper provides two main contributions.

1) To the best of our knowledge, ours appears to be the first work which recognizes that such a problem requires a novel formulation in the context of WLAN systems.

2) From the theoretical insights gained from the formulation of the model, we design, analyze, implement in a Linux Access Point, and experimentally assess, a *practical* (sub-optimal) scheduler whose performance are marginally lower than the optimal solution.

C. Related Work

A substantial amount of prior work applies SVC coding to video transmission over wireless networks. An high level framework and the general challenges of adaptive (scalable) video streaming in a wireless context are presented in [13]. Three techniques for video content delivery in such scenarios are identified: scalable video representation, an end-system capable of performing network aware adaptation (end-to-end approach), and adaptive QoS support from the network.

The representation of scalable video concerns the encoding of the video into different substreams with different quality levels as discussed in the standard H.264/SVC codec [14]. The network aware adaptation of end systems is often used to avoid congestion in the network, e.g., in conjunction with a transport protocol like TFRC [15]. A framework that uses TFRC for efficient congestion aware SVC video delivery has been proposed in [17]. Rate smoothness and real-time requirements for video streaming are addressed in [16]. Further proposals provide QoS support from the network, for example through layer-based in-network packet dropping [18], use of priority queuing taking into account the layers' importance [19], and rate distortion models for link adaptation [20].

None of the aforementioned solutions for SVC transmission over wireless networks consider multiple video streams, where the rate distortion properties of the videos depend on the specific video content of the individual streams. The fairness-throughput tradeoff when streaming multiple videos to different users has been analyzed in [21] based on a gradient-based scheduling, and in [22] based on remaining video playback time for each client. In [23], Ji et al. extend the gradient-based scheduling to optimize the resource allocation so as to meet delay constraints.

To date, most of the existing research is based on simulation and theoretical analysis, and few experimental results for SVC in WLAN are available [24]. Moreover, to the best of our knowledge, existing work does not consider the dependency of the MAC layer operation, even if in a centralized downstream setting, on the allocation of capacity to video transmissions. Furthermore, the impact of other applications as well as uplink traffic on the downlink scheduling problem has not been pointed out.

II. Utility-Maximizing Cross-Layer Downlink Scheduling Problem

In this section we formalize the problem of maximizing the utility of cross-layer scheduling for downlink video streaming over a WLAN network. The following is not restricted to video traffic, but addresses the general problem of maximizing the utility of generic downlink flows whose utility curve is known.

We consider an 802.11e WLAN formed by one AP delivering video streams to M associated stations. The video delivery occurs through the AP EDCA video access category. In addition, we consider further traffic (in a separate EDCA Access Category) generated at the stations and destined to the AP, that we call "non-video downstream" (nvd) traffic.

A. Assumptions and Notation

For simplicity, we use the following assumptions.

Assumption A1: the non-video-traffic is generated by a *constant* number of stations in saturation conditions [25], i.e., they always have a frame available for transmission.

Assumption A2: 802.11 frame transmissions are error-free, and collision of the video traffic generated by the AP with non video traffic generated by other stations is assumed negligible.

Assumption A3: the quality of each video stream i is described by a utility curve $U_i(b_i)$, where b_i is the average bit-rate assigned to stream i. The utility curves are assumed to be known at the AP for each considered stream.

Of these assumptions, A1 appears necessary to prevent the background traffic to further depend on the AP operation. A2 is realistic in the presence of a relatively small number of competing stations and considering the higher priority of the video access category. It is an assumption, which permits to neglect the complications that an analysis devised to further take into account channel collisions would raise. As such, it permits us to focus our contribution on the core aspects of the problem tackled in the paper. Finally, A3 restricts our treatment to utility curves based on *average* values. We have

specifically used average Peak Signal to Noise Ratio (PSNR) versus average application layer bit rate as utility metric, but any other utility metric based on average rates would fit our framework.

We use the following notation. Station i is the station which receives video stream i and we use index i to either refer to the receiving station or the delivered stream unless ambiguity occurs. We consider a multi-rate scenario where each station is connected to the AP using a specific PHY rate. Let C_i be the *maximum application-layer* delivery rate that is available to stream i, in the assumption that the AP might continuously transmit MAC frames to station i *without* any backoff time between consecutive frames. Clearly, C_i will be lower than the actual PHY rate assigned to the station, because of overhead, and is higher than the maximum throughput achievable by the stream. Simple computation allows to derive the value C_i from the PHY rates for a specific 802.11 PHY layer. For instance, for a station exploiting an 11 Mbps PHY rate, $C_i = 7.21$ Mbps, whereas $C_i = 1.76$ Mbps in the 2 Mbps case.

B. Problem Formulation

Our ultimate goal consists in determining and enforcing the combination of application-layer bit rates $b_1, .., b_M$ that the AP should grant to the M video streams so that the total utility $\sum_{i=1}^{M} U_i(b_i)$ is maximized under the constraint that the WLAN provides sufficient "channel resources" to deliver the resulting MAC layer traffic.

To formalize the problem we introduce variables x_i defined as the percentage of the *whole* WLAN channel time assigned to video stream i. The average application-layer bit rate b_i assigned to the i-th stream is $b_i = x_i C_i$. We conveniently classify the channel time into two categories:

- x_{vd}: percentage of time assigned to the AP for the transmission of MAC frames carrying video traffic;
- x_{nvd}: the remaining percentage of channel time.

The latter includes the time spent by the AP for independently delivering non video traffic through other EDCA AC queues, the time spent by *background* stations for their non video transmission, and the supplementary channel time wasted by the MAC protocol operation (specifically, unutilized channel time - empty channel slots - because of backoff counters count-down).

If x_{vd} were a known constant, the utility-maximizing allocation would be the solution of the straightforward constrained maximization problem:

$$\max_{\{x_1,..x_M\}} \sum_{i=1}^{M} U_i(x_i C_i) \qquad (1)$$
$$s.t. \quad \sum_{i=1}^{M} x_i = x_{vd}$$

We now proceed by deriving x_{vd}. Let a *round* be defined as the time interval between two consecutive AP transmissions of video packets. Figure 1 depicts the WLAN channel time as a sequence of consecutive rounds. The average duration of

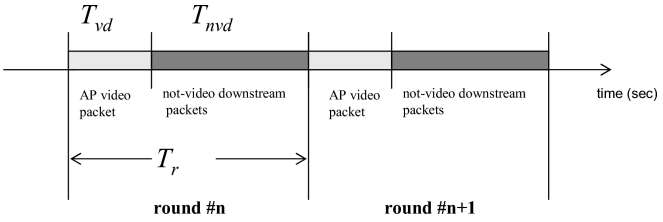

Fig. 1. Evolution of time as sequence of *rounds*

round T_r can be expressed as the sum of i) the average time T_{vd} consumed by the AP on the wireless interface to transfer a MAC frame containing video traffic, and ii) an average remaining time T_{nvd} which includes *both* the channel time wasted because of the WLAN MAC operation, as well as the time consumed for transmitting uplink or downlink non-video MAC frames. The percentage of time assigned to the AP for video traffic transmissions is then

$$x_{vd} = \frac{T_{vd}}{T_{vd} + T_{nvd}} \qquad (2)$$

Now, two fundamental observations hold:

1) Under the saturation assumption A1, T_{nvd} is a constant independent of the AP scheduling decision;
2) Conversely, the average time T_{vd} is directly affected by the specific scheduling decision, namely the choice of the tuple $\{x_1, ..x_M\}$.

As a consequence, x_{vd} is not constant but depends on the scheduling strategy. As an example, consider a single non-video station and let us assume that this station uses the same contention window of the AP's video access category. Given the same MAC parameters, in average a non-video transmission will occur for each frame transmitted by the AP (due to the long term fairness property of the 802.11 Distributed Coordination Function). Since i) the time to transmit the non-video frame depends only on the PHY rate of the non-video station, and since ii) the average number of channel slots that elapses between two consecutive channel transmissions is a constant which depends only on the MAC layer parameters ($CW_{\min}/4$, if we neglect collisions), also the average time T_{nvd} is constant and independent of the AP operation. In contrast, in a multi-rate scenario T_{vd} significantly depends on the choices made by the AP. If the AP transmits all frames to a station with a large PHY rate, this time will be shorter than in the case where the AP sends frames to a low PHY rate station.

Going back to the general case, let us redefine the scheduling rule as follows. Let α_i be the fraction of frames transmitted by the AP to station i with respect to the total number of frames transmitted by the AP, with the obvious constraint

$$\sum_{i=1}^{M} \alpha_i = 1 \qquad (3)$$

Then, the average time spent by the AP to transmit a video frame is computed as the weighted average

$$T_{vd} = \sum_{i=1}^{M} \alpha_i T_{x,i} \qquad (4)$$

where $T_{x,i}$ is the average time needed to transmit a frame to station i, depending on the rate of station i. Substituting equation (4) into (2),

$$x_{vd} = \frac{T_{vd}}{T_{vd} + T_{nvd}} = \frac{\sum_{i=1}^{M} \alpha_i T_{x,i}}{\sum_{j=1}^{M} \alpha_j T_{x,j} + T_{nvd}} \tag{5}$$

where we recognize that each addendum of this sum is indeed x_i, i.e.,

$$\frac{\alpha_i T_{x,i}}{\sum_{j=1}^{M} \alpha_j T_{x,j} + T_{nvd}} = x_i \tag{6}$$

The above considerations permit us to provide two equivalent formulations of the utility maximization problem in WLANs.

1) Formulation based on percentage of transmission opportunities α_i provided to MAC frames addressed to station i:

$$\max_{\{\alpha_1,\cdots,\alpha_M\}} \sum_{i=1}^{M} U_i(x_i C_i) \tag{7}$$
$$x_i = \frac{\alpha_i T_{x,i}}{\sum_{j=1}^{M} \alpha_j T_{x,j} + T_{nvd}}$$
$$s.t. \sum_{i=1}^{M} \alpha_i = 1$$

2) Formulation based on the percentage of channel time x_i allocated to the video stream i:

$$\max_{\{x_1,..x_M\}} \sum_{i=1}^{M} U_i(x_i C_i) \tag{8}$$
$$s.t. \sum_{j=1}^{M} x_j + T_{nvd} \sum_{i=1}^{M} \frac{x_i}{T_{x,i}} = 1$$

where the constraint yields from straightforward algebra. For an intuitive explanation of the latter constraint, consider a time interval of one second. Within a one second time period, x_i can be alternatively interpreted as the amount of time dedicated to stream i. Hence, $\sum_{i=1}^{M} x_i$ is the time consumed to transmit video downstream traffic. The remaining time is spent by the MAC backoff operation and by the transmission of non-video stations. This accounts, in average, to one time interval T_{nvd} per each frame transmitted by the AP. Since, in a second, the number of MAC frames delivered to station i is given by the ratio $x_i/T_{x,i}$, and hence the total part of the one second spent for MAC backoff operation and transmission of non-video frames is $\sum_{i=1}^{M} x_i/T_{x,i}$ multiplied by T_{nvd}.

From the utility maximization problem (8), we can identify the cross-layer information required to optimally schedule video traffic. The MAC layer information includes i) C_i for each $i = 1, ..M$, ii) the transmission time of each MAC frame $T_{x,i}$, and iii) T_{nvd}. The required application layer information is $U_i(b_i) = U_i(x_i C_i)$ for every delivered stream i.

III. Practical scheduler

In this section we design a practical scheduler, tailored to H.264 SVC, which leverages the insights emerged during the problem formalization, and conveniently exploits them in a form suitable for fast practical operation.

We remark that the optimal solution to the constrained maximization problem can be obtained for example using linear programming techniques. However, the problem solution require the run-time estimation of T_{nvd} (that can be also complicated) and the consequent change of scheduler policies. To overcome this problem we propose in this section a practical scheduler, that is T_{nvd} independent, but that can lead the performance close to the optimum.

The goal of the practical scheduler is to guarantee that excess NALUs are dropped in agreement with the H.264 decoding dependencies thus we first briefly provide essential information on how they are enforced in the H.264 SVC standard [6].

A. H.264 SVC background

An H.264 SVC stream is a sequence of NALUs. A NALU is formed by an 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 [6], or [10]), 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*, namely the ability to adapt the video spatial resolution (e.g., from CIF to 4CIF). TID allows *Temporal Scalability*, i.e., it provides the ability to adapt the video frame-rate. For instance, if a stream is coded with a frame rate 30 frames/second, by dropping all NALUs marked with the higher TID value we achieve a 15 frames/second decoded video, and so on. Finally, QID allows *Medium Grain Scalability*, also called *progressive refinement*). Each NALU received with a QID parameter greater than 0 adds supplementary quality to the spatial/time substream (i.e., DID and TID parameters) the NALU belongs to. Specifically, each additional quality layer reduces the encoding quantization error, improving PSNR.

For our purposes, it is essential to drop excess NALUs so that decoding dependencies are respected. Restricting our attention to the temporal and medium grain scalability only (for simplicity, in our experimental results we have not considered spatial scalability), the following decoding dependencies hold (the arrow means "depends on"):

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

The first rule states that temporal dependencies are enforced only on the quality layer 0, i.e., that a NALU belonging to the temporal-layer $tid > 0$ and with $qid = 0$ depends on NALUs of temporal-layer $tid - 1$, again with $qid = 0$. The second rule states that quality improvements are progressively applied to a considered temporal layer, i.e., a NALU belonging

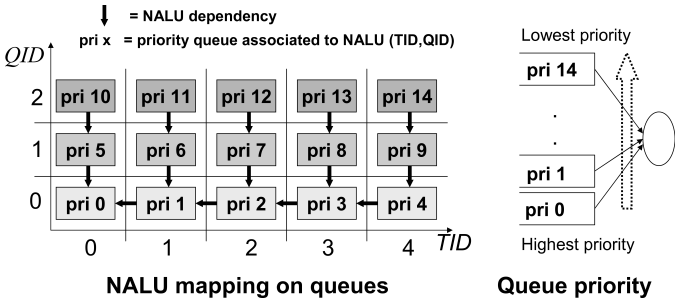

Fig. 2. mapping of SVC substreams to priority queues

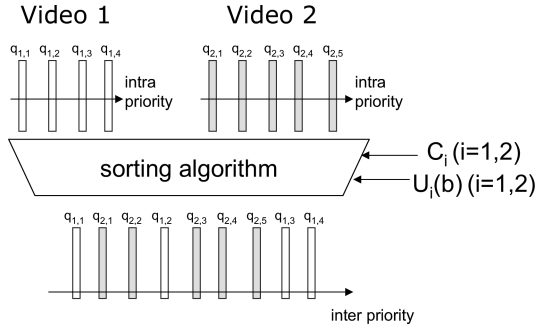

Fig. 3. Conceptual sketch of queue merging

to the quality-layer $qid > 0$ depends on NALUs of quality-layer $qid - 1$, with the same DID and TID. These rules are graphically highlighted in the left drawing in figure 2.

B. practical scheduler overview

The practical scheduler that we propose runs *above* the MAC layer, and it is devised to properly arrange the order of NALUs delivered to the MAC layer. This permits to retain a fully standard and application-layer unaware MAC operation. As discussed below, the priority order depends on cross-layer information, and specifically it depends on both utility information provided by the application layer, as well as per-station channel rate information provided by the MAC layer (gathered by the NIC driver). A fundamental feature of our proposed approach is that, unlike the optimal approach presented in the previous section, it does not require the explicit *run-time* knowledge of the T_{nvd} time, indeed an information not easily gathered from the NIC driver. As a result, the proposed approach is seamlessly adaptive to any available AP delivery capacity and related fluctuations. Of course, this simplicity is paid with a sub-optimal operation; however, numerical results will later on prove that, with average PSNR versus rate utility curves, the performance degradation is almost negligible.

A convenient way to order NALUs as well as drop excess ones is to rely on a bank of small-size (we used 10 NALUs each) priority queues . The assignment of NALUs to queues, as well as priority values to queues, must take into account both i) the requirement that NALU decoding dependencies must be respected in the delivery order, as well as ii) the goal of maximizing the utility brought by the delivered NALUs. As illustrated in figure 3, we propose to accomplish this goal by:

1) deploying a number of dedicated queues per each video stream, so that each queue carries the stream NALUs belonging to a specific video layer for that stream, and arrange them in an *intra-stream priority order* (section III-C);
2) sorting the deployed queues on the basis of cross-layer information (utility and rate per each video stream), so that they are arranged in an *inter-stream* priority order (section III-D);
3) deploying a *flat service priority discipline* for orderly draining NALUs from the sorted bank of queues.

C. intra-stream priority

The traffic generated by a same video stream is conveyed to a bank of queues, where each queue accommodates NALUs belonging to a given scalability layer, i.e. a (TID,QID) pair. 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, numbered from 0 to 14.

Intra-stream queues are sorted by setting a priority order among the video layers. As discussed in [24], a natural approach is to use the video-layer-to-priority mapping illustrated in figure 2, which gives higher priority (0 being the highest priority) to the temporal scalability layers (in their TID order), and then decreasing priority to the corresponding quality enhancement. In formulae, a NALU with $QID = q$ and $TID = t$ is delivered to the queue with priority index $s = 5q + t$.

We remark that this intra-stream priority assignment does respect the decoding dependencies enforced by H.264 SVC and permits to reach a transfer efficiency (one minus the percentage of NALUs received at the destination, but discarded because of missing dependencies) very close to 100% [24].

D. inter-stream priority

Let us denote with $q_{i,s}$ the queue of the video stream i that handles NALUs for the video substream $s = 5q + t \in (0, 14)$. Let $b_{i,s}$ be the average application-layer bit rate rate needed to deliver all the i-th video substreams until level s. Let $\{U_i(b_{i,s})\}$ be a vector of utility values computed in correspondence of the rates $b_{i,s}$, i.e. the utility brought by the transmission of all the video layers until level s.

The inter-stream queue sorting is based on a greedy algorithm. Rather than optimizing the scheduling for a given T_{nvd} value, the algorithm scans a set of pre-established quantized T_{ndv} values (or decision points). In our algorithm, T_{nvd} values are scanned in decreasing order, since a lower T_{nvd} value yields an higher AP capacity for delivering video traffic. For each T_{nvd} value, we determine the next queue to be served, namely the one which provides the best possible utility improvement. The sub-optimal nature of the proposed algorithm stems from the fact that the queue ordering committed for a prior value T_{ndv} can only be extended with a next queue, but not changed.

The following three parameters ($step$, N_{steps}, U_{th}) are used to perform the greedy algorithm:

- $step$. This is the time step according to which the T_{nvd} values are quantized. We used a small step (1 ms, roughly equivalent to the transmission time of a MAC frame at maximum 802.11b PHY rate), so that at each step, at most one new queue is accommodated in the sorted list;
- N_{steps}. A large value (we used 200), so that the starting value $T_{nvd} = N_{steps} \cdot step$ does not permit to accommodate any stream;
- U_{th}. A minimum utility increment in order to take a commit decision (we used 0.5 dB for PSNR utility curves).

In brief, at each decision point, the greedy algorithm works as follow:

1) At the decision point h, we compute the total application-layer bit rate $b_i(h)$ that the i-th video would receive if all the queues up to now inserted in the inter-stream priority list were completely served. From $b_i(h)$ we derive the corresponding time percentage $x_i(h) = b_i(h)/C_i$.
2) We then check whether the next decision point $h-1$, corresponding to a smaller T_{ndv} value, would permit to accommodate a new queue. This is accomplished by computing the stability constraint condition in the formalization (8) as:

$$sc(h-1) = \sum_{i=1}^{M} x_i(h) + \left(\sum_{i=1}^{M} \frac{x_i(h)}{Tx_i} \right) T_{nvd}(h-1) \qquad (9)$$

If the value $sc(h-1)$ is lower than 1, extra space is available for accommodating an additional queue.
3) Provided that this is the case, we compute, for each stream, what would be the utility improvement if we pick the latest not yet inserted queue for that stream as next queue to be served in the inter-stream priority order illustrated in figure 3. This is provided by computing, for all streams i, the quantities

$$\tilde{x}_i(h-1) = \frac{1 - \left(\sum_{j=1, j\neq i}^{M} x_j(h) + \sum_{j=1, j\neq i}^{M} \frac{x_j(h) T_{nvd}(h-1)}{Tx_j} \right)}{1 + T_{nvd}(h-1)/Tx_i}$$
$$I(h-1|i) = U_i(\tilde{x}_i(h-1)C_i) - U_i(x_i(h)C_i) \qquad (10)$$

where $\tilde{x}_i(h-1)$ is the percentage of time assigned to stream i if all the new capacity available is provided to such stream, and $I(h-1|i)$ is the utility improvement, with respect to the overall utility achieved in the previous iteration, that we would obtain if all the extra-time were given to the i-th stream.
4) Finally, we select, as the next queue to be inserted in the list, the one which yields the maximum utility improvement $I(h-1|i)$, provided that such an improvement is greater than a predetermined threshold U_{th}.

The procedure of the algorithm is elaborated in algorithm 1.

Algorithm 1 inter-stream queue sorting

Input: Utility function U, number of video M, increase of step size $step$, minimum expected utility change U_{th}, number of decision points N_{step}.
Output: Optimal sequence of priority queues $\tilde{q}_{opt}$;
Initialization: index of last queue of the video i-th inserted: $last_i = 0$, index of decision point: $h = N_{step}$, Iteration index, $I = 0$.
while there are queues not yet inserted **do**
  $T_{ndv}(h-1) = (h-1) * step$
  for $i = 1$ to M **do**
    $b_i(h)$ = cumulative bitrate of the i-th video up to substream $last_i$
    $x_i(h) = b_i(h)/C_i$
  end for
  $sc(h-1)$ = see eq. 9
  if $sc > 1$ **then**
    continue while loop
  end if
  $max\_index = 0$
  $max\_value = 0$
  for $i = 1$ to M **do**
    if $last_i ==$ number of i-th video substreams **then**
      continue for
    end if
    $\tilde{x}_i(h-1)$ = ... see eq. 10
    $I(h-1|i)$ = ... see eq. 10
    if $I(h-1|i) > max\_value$ **then**
      $max\_value = I(h-1|i)$
      $max\_index = i$
    end if
  end for
  if $max\_value > U_{th}$ **then**
    $i = max\_index$
    $k = last_i + 1$
    $last_i = last_i + 1$
    $flat_q \leftarrow q_{i,k}$ #insert the new queue related to video i-th and substream k-th
  end if
  $h = h - 1$
end while
output: $\tilde{q}_{opt}$

E. Analytical Utility performance of the proposed algorithm

Given a value T_{ndv}, is possible to analytically compute the utility performance provided by the proposed algorithm. It suffices to compute the rate b_i granted by the scheduling algorithm's operation to each i-th video stream and sum the corresponding utilities.

For this purpose, let us enumerate the queues according to their assigned priority order, from 1 (higher priority queue) to W. For a given T_{nvd} value, the priority scheduler will serve the set of queues $1..lsq$, where $lsq \leq W$ is the index of the last queue served by the scheduler. The bitrate b_i granted by the scheduler to the i-th video is the cumulative bitrate of the streams of the i-th video that feed the $1..lsq$ queues. This latter evaluation of b_i is straightforward, since the association queue-stream is known. Thus, the only remaining problem is the determination of lsq for a given T_{nvd} value. The iterative approach detailed in algorithm 2 is designed for this purpose. We start considering the first priority queue and iteratively add a queue, following the priority order. At each iteration, we evaluate the cumulative bitrate b_i associated to each video, given that all the considered queues are fully drained. From b_i we evaluate the time percentage x_i and verify the stability constraint. Whenever the stability constraint first exceeds one, it means that the last added queue is precisely the lsq one. The bit rate assigned to the lst queue is finally determined by computing, through the stability constraint, the percentage of time x_{lsv} assigned to this queue and consequently derive b_{lsv}.

Algorithm 2 Calculate $U_{tot-sub}$

for $j = 1$ to W **do**
  for $i = 1$ to M **do**
    $b_i =$ cumulative bitrate of the i-th video feeding the queues $1..j$
    $x_i = b_i / C_i$
  end for
  $sc = \sum_{i=1}^{M} x_i + \left(\sum_{i=1}^{M} \frac{x_i}{Tx_i} \right) T_{nvd}$
  if $sc > 1$ **then**
    $lsq = j$
    $lsv =$ index of video associated with lsq break the FOR
  end if
end for
$x_{lsv} = \frac{1 - \left(\sum_{j=1, j \neq i}^{M} x_j + \sum_{j=1, j \neq ldv}^{M} \frac{x_j T_{nvd}}{Tx_j} \right)}{1 + T_{nvd}/Tx_{lsv}}$
$b_{lsv} = x_{lsv} C_{lsv}$ and $U_{tot-sub} = \sum_{j=1}^{M} U_i b_j$

F. Linux implementation

We conclude the section by briefly discussing the implementation of the practical scheduler in Linux. We employed an AP equipped with an Atheros card. We modified the MadWiFi driver in order to run-time retrieve both the number of empty spaces available in the EDCA AC video queue, and the actual PHY rates employed by the AP to transmit MAC frames to each station i. The scheduler has been implemented in the Linux OS through the Linux traffic control tool at the IP layer. To prevent from losses emerging because of NIC buffer overflow, we have developed a simple flow control mechanism for delivering frames from the scheduler to the driver. It is implemented by periodically (250 ms) sampling the NIC buffer availability, and setting the IP queues draining rate for the next period to the maximum rate which guarantees that no MAC buffer overflow will occur even if the AP will not transmit at all in the next period. With the same periodicity, we query the driver for retrieving information about changes in the PHY rate of the connected stations. If this occurs, we rerun the algorithm and change the inter-stream queue priorities accordingly.

IV. Performance evaluation

We have evaluated the performance of the practical scheduler over an 802.11e WLAN test-bed. The experimental assessment of H.264 SVC in-network adaptation approaches is not straightforward, and it was made possible only by employing the tools and the methodology recently introduced in our companion work [12], and made available as public domain open source library at http://svef.netgroup.uniroma2.it. Due to space restrictions, we refer the reader to these references for supplementary details.

A. Experimental Scenario and Utility Function Evaluation

We used the soccer video with 4CIF resolution and 30 frames/s as a reference. We coded it as H.264 SVC stream using the Joint Scalable Video Model reference software (JSVM, [27]). The resulting coded video comprises a base layer, including 5 temporal scalability layers, plus two Medium Grained Scalability (quality enhancement) layers per each temporal layer, for a total of 15 video substreams. For the coded video, we off-line computed an utility curve described in terms of average PSNR versus average bit rate. The computation of the PSNR versus rate curve was performed by stripping out the layers, measuring the average bit rate, decoding the resulting video, and computing the average PSNR. The resulting PSNR curve is reported in figure 4.

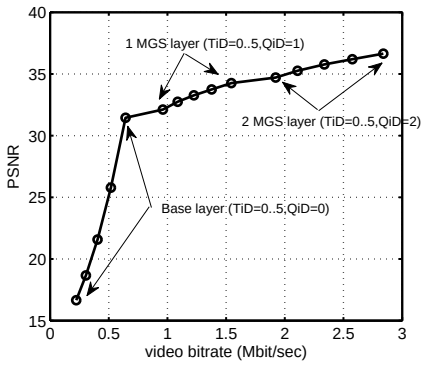

Fig. 4. PSNR versus bitrate of the coded video used in the analysis

The considered network scenario comprises of a number of downlink video streams addressed to different stations, and generated from the same video sequence translated in time. In addition to the video downstreams, we generate non video traffic from a variable (from 0 to 5) number of stations transmitting at 2 Mbit/sec, which uploads UDP greedy traffic (saturation conditions) generated through iperf. The remaining stations are Linux laptops with Ralink WiFi chipsets. The uplink traffic is best effort, thus the Ralink driver is set with Cwmin=31 and AIFS=2. On the contrary, the video traffic accesses the channel with Cwmin=15 and AIFS=1.

B. Experimental and theoretical performance versus the Number of Uplink Stations

In these experiments we consider two video streams delivered by the AP to two different stations, one connected with an 11 Mbps PHY rate, and the other with a driver-enforced 2 Mbps PHY rate. To prove the practical scheduler effectiveness, we introduced in the wireless network other N_{up} greedy stations that transmit packets at 2Mbps in the uplink direction. Changing the number N_{up} we induced in the wireless network different T_{nvd}. The downlink video traffic is delivered using the WME/EDCA video queue and the uplink traffic transmitted by the other stations uses best-effort queue.

Performance are measured in terms of cumulative utility (PSNR) delivered to the two video receivers and are plotted versus the number of uplink greedy stations, N_{up} that is varied from 0 to 5. Note that the actual value of T_{nvd} for each experiment is independent on the scheduler and is only a function of the PHY bit rate and of the number of greedy stations, N_{up}. Thus the optimal solution has to be computed for each N_{up} value. On the contrary the practical scheduler works independently on T_{nvd} value.

A fundamental problem we needed to face is the need to derive the average value T_{nvd} that resulted for each specific value of N_{up} to compare the measured results with the analyitcal. We were not able to derive such an average

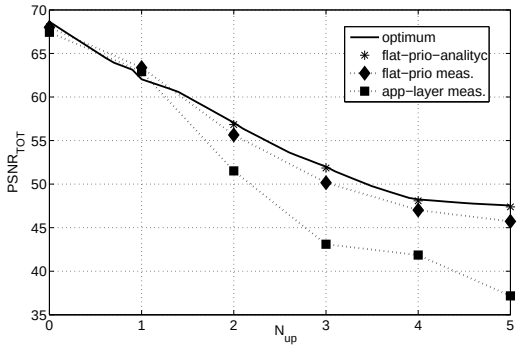

Fig. 5. Cumulative PSNR verus the number of Uplink Stations

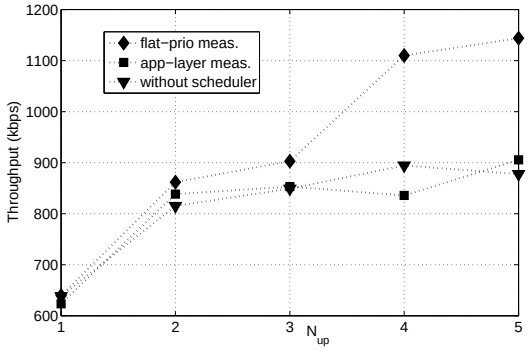

Fig. 6. Aggregated throughput of uplink stations

value from driver-level information. Moreover, the analytical derivation of such a parameter, in principle relatively easy (e.g., using EDCA extensions of the model [25]) from the knowledge of the MAC layer parameters employed by the competing stations, was discouraged by the fact that, as shown in [26], the operation of the two considered cards slightly, but noticeably for our specific purposes, differs from the theoretically expected performance.

Therefore, we resorted to an hybrid experimental/analytic off-line estimation of the T_{nvd} parameter. At first, we experimentally derived the actual AP throughput ρ_{AP} achieved when competing with k background best-effort Ralink stations using the 2 Mbps PHY rate (as in our scenario); All stations were loaded with saturated UDP traffic with 1500 bytes IP packets. Then, we derived T_{nvd} by recognizing that $T_{nvd} = \frac{1500 \cdot 8}{\rho_{AP}} - T_{x,AP}$, being $T_{x,AP}$ the computed transmission time for a 1500 bytes MAC frame by the AP. Using this approach, the obtained values of T_{nvd} for the different values of N_{up} are reported in Table I.

Figure 5 reports the cumulative PSNR as resulting from i) the solution of the optimal scheduling rule determined by solving equation 8 (marked as "optimum"); ii) the analytical results obtained by analytically computing, through the procedure 2, the utility achieved by the proposed sub-optimal flat priority scheme detailed in alg. 1 (marked as "flat prio analytic"); iii) the experimental results obtained by running the Linux AP implementation of the proposed flat priority approach (marked as "flat prio meas."); iv) the measurements obtained implementing a scheduler which does not rely on the knowledge of the PHY rate at which stations are connected, but uses only application layer information (marked as "meas. app. layer sched"). This scheduler deploys 15 queues, one per each video layer. However, with no supplementary insights on the available PHY rate at which streams are connected, and in the considered scenario where video layers bring the same utility for different streams (we recall that we used the same video sequence for all streams), its best strategy is simply to share the queues among the different video streams and drain traffic from the queues according to their intra-stream priority order. We have also measured the performance without any scheduler. The results have not been shown because the maximum of the curves was $29.13dB$ of PSNR, corresponding to the case without uplink traffic ($N_{up} = 0$), and thus the readability of the figure would have been compromised.

From figure 5, we see that the performance advantage of the optimal scheduler is negligible with respect to the analytical solution of the practical scheduler, and it is marginal also when compared with the actual experimental results.

It could be argued that the suboptimality of the practical scheduler may depend on the considered utility curve, and that it is possible to find cases where the performance difference becomes notable. An in depth analysis of the impact of different utility shapes over the performance is left to further work. However, we believe that major performance differences are deemed to emerge only with very particular, and unrealistic, utility shapes. As expected, comparison with the application-aware-only scheduler shows that the usage of cross-layer information yields a significant performance improvement, especially in the case of significant capacity restriction (large values of T_{nvd}).

Finally, in figure 6 we report the aggregated non-video throughput for the greedy N_{up} uplink stations. A positive side effect of our proposed approach is that not only it does not penalize uplink stations, but it may even improve the uplink traffic throughput, as comparison with the application-layer-only scheduler and the absence of scheduler shows. This apparently counter-intuitive fact can be explained by considering that, in order to maximize the Utility, the crosslayer scheduler tends to select packets directed to the 11 Mbps stations, hence increasing the overall network throughput of the wireless network (in other words, the scheduler tends to limit the WLAN performance anomaly).

TABLE I
VALUES OF T_{nvd} FOR N_{up} GREEDY STATIONS AT 2 MBPS PHY RATE

N_{up}	1	2	3	4	5
T_{nvd}	0.004439	0.005874	0.007725	0.0102	0.0116

C. Experimental and theoretical performance versus number of video downstreams

A second set of measurements was performed to determine the scheduler operation for a varying number of video streams. Figure 7 shows results obtained in a scenario characterized by one background best-effort station transmitting at 2 Mbps, and a varying number of video streams, of which one is delivered at 2 Mbps, and the remaining at 11 Mbps.

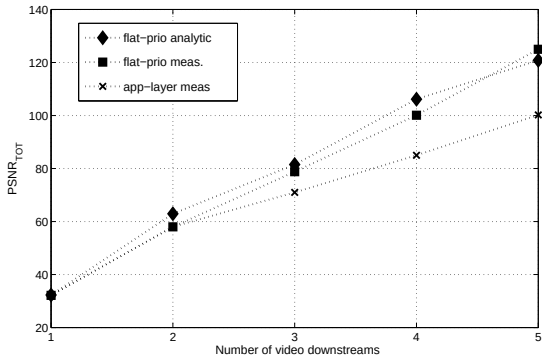

Fig. 7. Cumulative PSNR versus number of video downstreams

The figure compares the experimental and analytic results obtained by the "flat prio" scheduler, with that of the application-layer-only approach. The results for the optimal scheduler are perfectly coincident, for such scenario, with that of the proposed practical "flat prio" approach. As expected, the figure confirms the superiority of the cross-layer approach with respect to an application-aware-only approach.

The growth of the total utility with an increased number of video streams may not be considered intuitive (indeed, a comparable shared capacity is provided by the AP), but it is readily explained by considering that, as shown in figure 4, the relative utility gain is large at low bit rates, and gets progressively smoother as more bit rate is provided to a given stream (as in the case of a small number of video stream).

V. Conclusion

The paper addresses the issue of delivering scalable video over Wireless LANs. A major contribution of the paper consists in the formalization of such problem, taking into account the unique characteristics of the WLAN MAC operation. Actually, two equivalent formulations are provided, one more suitable for a scheduler implemented at the MAC layer, the other one more suitable for an IP layer implementation.

Leveraging the theoretical insights gained from the formulation of the model, we have designed, analyzed, and implement in a Linux Access Point a *practical* (sub-optimal) scheduler. Moreover, we have experimentally assessed its performance over a real world WLAN trial and using PSNR-based utility curves extracted from video sequences. The performance of the proposed practical scheduler are shown to be only marginally lower than the optimal solution.

References

[1] IEEE Standard 802.11-2007, Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, June 2007.

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

[3] 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.

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

[5] M. van der Schaar, S. Krishnamachari, Choi Sunghyun, Xu Xiaofeng, "Adaptive cross-layer protection strategies for robust scalable video transmission over 802.11 WLANs", IEEE Journal on Selected Areas in Communications, Volume 21, Issue 10, Dec. 2003, pp. 1752-1763

[6] ITU-T recommendation H.264: Advanced video coding for generic audiovisual services, International Telecommunications Union, Nov. 2007.

[7] 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

[8] 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

[9] I. Kofler, M. Prangl, R. Kuschnig, H. Hellwagner, "An H.264/SVC-based adaptation proxy on a WiFi router", ACM NOSSDAV, Braunschweig, Germany, May 2008, pp. 63-68.

[10] R. Kuschnig, I. Kofler, Michael Ransburg, H. Hellwagner, "Design options and comparison of in-network H. 264/SVC adaptation", J. Visual Commun. and Image Repres., Dec. 2008, vol. 19, pp. 529-542.

[11] 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 International Conference on Visual Information Engineering, Aug. 2008. VIE 2008, pp.723-728

[12] A. Detti, G. Bianchi, C. Pisa, S. Proto, P. Loreti, W. Kellerer, S. Thakolsri, J. Widmer, "SVEF: an Open-Source Experimental Evaluation Framework for H.264 Scalable Video Streaming", MediaWin, Sousse, June 2009 - Software available at http://svef.netgroup.uniroma2.it

[13] D. Wu, Y. T. Hou, Y.-Q. Zhang, "Scalable video coding and transport over broadband wireless networks", Proc. IEEE, vol. 89, pp. 6-20, 2001.

[14] H. Schwarz, M. Wien, "The Scalable Video Coding Extension of the H.264/AVC Standard," IEEE Signal Processing Magazine, vol. 25, no. 2, pp. 135-141, Mar. 2008

[15] M. Handley, S. Floyd, J. Padhye, J. Widmer, "TCP Friendly Rate Control (TFRC): protocol specification", RFC3448, Jan. 2003. Available: http://www.ietf.org/rfc/rfc3448.txt

[16] J. Vieron, C. Guillemot, "Real-time constrained tcp-compatible rate control for video over the internet", IEEE Trans. Multimedia, vol. 6, no. 4, pp. 634-646, 2004.

[17] O. Hillestad, A. Perkis, V. Genc, S. Murphy, J. Murphy, "Adaptive H.264/MPEG-4 SVC video over IEEE 802.16 broadband wireless networks", Packet Video 2007, Lausanne, CH, Nov. 2007, pp. 26-35.

[18] T. Schierl, T. Stockhammer, T. Wiegand, "Mobile Video Transmission using Scalable Video Coding (SVC)", IEEE Trans. on Circuits and Systems for Video Technology, June 2007.

[19] Hsing-Lung Chen, Po-Ching Lee, Shu-Hua Hu, "Improving Scalable Video Transmission over IEEE 802.11e through a Cross-Layer Architecture", Fourth Int. Conf. on Wireless and Mobile Commun, 2008, pp. 241-246.

[20] Y. P. Fallah, H. Mansour, S. Khan, P. Nasiopoulos, H. M. Alnuweiri, "A Link Adaptation Scheme for Efficient Transmission of H.264 Scalable Video Over Multirate WLANs", IEEE Trans. on Circuits and Systems for Video Technology, Vol. 18, No. 7, Jul 2008, pp. 875-887.

[21] R. Agrawal, V. Subramanian, "Optimality of certain channel aware scheduling policies", Proc. of 2002 Allerton Conference on Communication, Control and Computing, 2002.

[22] T. Ozcelebi, M.O. Sunay, M.R. Civanlar, A.M. Tekalp, "Application-Layer QoS Fairness in Wireless Video Scheduling", IEEE International Conference on Image Processing, 8-11 Oct. 2006, pp. 1673 - 1676.

[23] X. Ji, J. Huang, M. Chiang, G. Lafruit, F. Catthoor, "Scheduling and resource allocation for SVC streaming over OFDM downlink systems", IEEE Trans. on Circuits and Systems for Video Technologies, 2009.

[24] G. Bianchi, A. Detti, P. Loreti, C. Pisa, F. S. Proto, W. Kellerer, S. Thakolsri, J. Widmer, "Application-aware H.264 Scalable Video Coding delivery over WLANs: Experimental Assessment", IEEE IWCLD, June 2009, Palma de Mallorca.

[25] G. Bianchi, "Performance Analysis of the 802.11 Distributed Coordination Function", IEEE Journal of Selected Areas in Commun. Vol. 18, N. 3, March 2000, pp. 535-547

[26] G. Bianchi, A. Di Stefano, C. Giaconia, L. Scalia, G. Terrazzino, I. Tinnirello, "Experimental Assessment of the Backoff Behavior of Commercial IEEE 802.11b Network Cards", IEEE Infocom 2007, May 2007, pp. 1181 - 1189.

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