Ad Hoc Networks 24 (2015) 74–84

Contents lists available at ScienceDirect

Ad Hoc Networks

journal homepage: www.elsevier.com/locate/adhoc

# Performance evaluation of sender-assisted HTTP-based video streaming in wireless ad hoc networks

Stefania Colonnese [a,*], Francesca Cuomo [a], Raffaele Guida [a], Tommaso Melodia [b,1]

[a] University of Roma Sapienza, Department of Information Engineering, Electronics and Telecommunications (DIET), Via Eudossiana 18, 00184 Roma, Italy
[b] Northeastern University, Department of Electrical and Computer Engineering, Boston, MA 02115, USA

ARTICLE INFO

*Article history:*
Received 13 March 2014
Received in revised form 2 July 2014
Accepted 28 July 2014
Available online 7 August 2014

*Keywords:*
HTTP adaptive streaming
Wireless ad hoc networks
Quality of Experience (QoE)

ABSTRACT

HTTP Adaptive Streaming (HAS) delivers video streaming services according to a client–server architecture where the client originates consecutive HTTP requests to download chunks of encoded video. In state-of-the-art systems, the client selects the chunk out of a finite set of differently encoded versions of the same video made available at the server site; the selection is driven by a client-centered buffer management procedure. Still, dynamic bitstream switching may have drawbacks in terms of undesirable visual quality fluctuations artifacts at the final user; besides, it may result in oscillatory behavior of the overall traffic in case of multiple users. Therefore, this paper proposes a sender-assisted procedure for HTTP Adaptive Streaming (HAS) services with improved user Quality of Experience (QoE) that proactively avoids buffer underflow events at the receiver side, thus reducing the need for dynamic bitstream switching. In the proposed sender-assisted approach, HAS leverages information on the encoded video available at the server side to assist the client in originating the data requests. Specifically, the sender-assisted HAS procedure exploits information on the encoded video content available at the sender side to regulate the interval between consecutive client-originated download requests. Significant QoE improvements brought by the proposed sender-assisted video streaming procedure are demonstrated in challenging fluctuating throughput conditions encountered in wireless ad hoc networks.


## 1. Introduction

The demand of video streaming services is rapidly increasing and is expected to unrelentingly grow in the next few years [1]. Mobile and wired network traffic is composed of an ever increasing percentage of video streaming traffic and the efficiency of streaming services will certainly impact the resulting network load. The most popular streaming architecture, known as HTTP Adaptive Streaming (HAS), underlies several commercial streaming video services in current video networking systems. HAS is used in world-wide adopted streaming platforms such as Microsoft's Smooth Streaming, Adobe's HTTP Dynamic Streaming, or Apple's HTTP Live Streaming [2].

HTTP-based systems benefit from full compliance with network devices and middleboxes (e.g., firewalls, NATs) currently used in the Internet. Adaptivity allows the client to select video chunks out of a set of bitstreams encoded at different quality levels and representing the same video and made available for bitstream switching purposes at the server side. In this way, the download rate of the chunks can be adapted to the current network conditions (i.e., available bandwidth). Therefore, the effective video streaming rate depends on both the encoding settings

* Corresponding author. Tel.: +39 0644585643; fax: +39 064744481.
*E-mail addresses:* colonnese@infocom.uniroma1.it (S. Colonnese), francesca.cuomo@uniroma1.it (F. Cuomo), raffaeleguida1984@gmail.com (R. Guida), melodia@ece.neu.edu (T. Melodia).
[1] The work of Tommaso Melodia was supported by the US National Science Foundation under grant CNS-1117121.

http://dx.doi.org/10.1016/j.adhoc.2014.07.023
1570-8705/

and on the end-to-end channel throughput experienced between the sender and the client [3]. Being based on the underlying reliable TCP network protocol, HAS prevents packet error and losses at the price of introducing random end-to-end packet transmission delays, which in turn may result in underflow events at the client buffer. A standard framework for HAS, known as Dynamic Adaptive Streaming using HTTP (DASH) [4], has also been defined within MPEG and 3GPP standardization activities.

When an underflow event occurs, video playout is interrupted and the client enters a rebuffering phase until the buffer is loaded to a preset threshold prior; then, playout restarts. Underflow events and rebuffering delays affect the quality perceived by the user, especially on wireless IP networks with large TCP throughput variations [5]. Even in stable end-to-end bandwidth conditions, random delay is introduced because of the random size of the encoded video chunk [6]. Therefore, a significant margin between the current throughput and the average encoding rate is required to avoid non-negligible QoE degradation due to fluctuations of the encoding rate. Furthermore, experimental evaluations of different commercial HAS systems [2,7] show that even the visual quality fluctuations caused by the client-based rate adaptation strategies result in fluctuations of the Quality of Experience (QoE) at the client side.

In this paper, we propose a sender-assisted HTTP streaming strategy that avoids unnecessary dynamic rate adaptation using an efficient scheduling of the HTTP-GET request underlying the streaming session. Specifically, the sender (i) proactively foresees upcoming underflow events using information on the encoded video content available at the sender side, and (ii) it uses this information to adaptively control the duration of the intervals between consecutive video segment requests performed by the client. The proposed strategy enables efficient and flexible management of the rapidly varying random fluctuations of the packet delays that are caused by the interplay of the characteristics of the encoded content and of the network conditions. This approach is especially beneficial in wireless ad hoc networks where challenging network conditions are typically encountered as a consequence of the variability of network accesses and of the wireless channel and interference dynamics.

The remainder of this paper is organized as follows. Section 2 discusses related work. Section 3 introduces the architecture of the discussed sender-assisted video streaming procedure, while Section 4 proposes a model of its performance. Section 5 discusses the proposed buffer management algorithm. Section 6 discusses performance evaluation results, and Section 7 concludes the paper.

## 2. Related work

Video streaming in wireless systems has been addressed in several papers, e.g., [8–11]. A cross-layer selection of the optimal parameter values that maximize the expected user-perceived video quality is proposed in [8]. This requires computing, for each user and each parameter set, the expected video reconstruction quality at the receiver application layer. The proposed approach is shown to lead to good performance even if the computational cost is high and dramatically increasing with the number of users. In [9], the authors analyze the video bitstream across a multi-hop IEEE 802.11a/e wireless network by investigating (i) the video quality improvement that can be obtained with an integrated cross-layer optimization involving all layers of the protocol stack and (ii) the impact on performance and complexity if the optimized streaming solution is performed using only limited, localized state information. However, the authors consider the case of a general video encoded reproducing the MPEG behavior.

In [10], the authors present a buffer and rate optimization algorithm aimed at reducing frame loss, buffer underflow events, and buffer delay in wireless networks, so as to improve the users' QoE. This algorithm applies a low-cost bandwidth estimation approach at the application layer to provide information on the effective capacity, available bandwidth, and variance in bandwidth for the bottleneck wireless link. Different from our approach, the authors propose to employ, before starting video streaming, a mechanism to capture the time-varying nature of the current wireless conditions within a cumulative density function (CDF) estimate. Conversely, we rely only on an approximation of the buffer occupancy obtained by measuring the current wireless access behavior (in terms of MAC collisions) and size of video chunks sent to the client.

Another class of papers deals with mobile video adaptation. As an example, in [12] challenges related to the variability of TCP throughput in wireless networks as well as to the management of client-side video playback buffer and to variations of the battery level of the mobile terminals are faced. The authors propose an adaptation algorithm consisting of one module that derives the rate for chunk downloading based on the current buffered video time, recent TCP throughput history, and video rates for the previous chunks and one module to generate bundled chunk download schedule to increase the energy efficiency of 3G radios. Similar to this approach, we propose a server-side adaptation algorithm suitable for ad hoc networks where different competing communications share the channel. In our case, the target performance metrics are related to the buffer freezes and we define a content-aware approach for mobile HAS that regulates client HTTP requests.

We propose a short-term rate adaptation scheme based on a look-ahead mechanism operated at the sender side. We expect the proposed scheme to be able to reduce playout buffer underflows and oscillations that characterize the system in Fig. 1. Within this context, the objective of this work is to propose enhancements to state-of-the-art HTTP-based video streaming protocols specialized for ad hoc networks, where mobile devices may experience large bandwidth fluctuations due to either physical layer fluctuations of the wireless channel or to random MAC delays due to coexistence of multiple users. The proposed HAS architecture provides an effective basis to develop a new generation of flexible HTTP video streaming strategies that can match these specific challenges of ad hoc networks.

## 3. Sender-assisted HAS architecture

Typical HAS systems, e.g., Dynamic Adaptive Streaming based on HTTP (DASH) systems [4], are characterized by the consecutive stages described in the following. First,

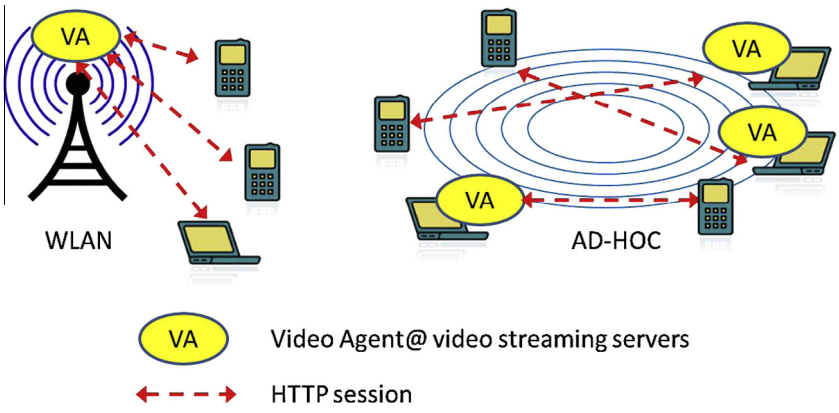



**Fig. 1.** Architectural model.

the video content is encoded at the server side in multiple versions, each with a different video quality. Each encoded bitstream is then parsed into video packets, usually referred to as "chunks", which encompass one or more GOPs (Group of Pictures), and are addressed by means of URLs available to the client through HTTP servers. At the beginning of the session, the server advertises the characteristics of the encoded video to the client via a Multimedia Presentation Description (MPD) message. The contents are then selected by the client and fetched in the form of chunks by means of HTTP GET requests. In HAS, chunking and storing may be realized by segmentation or fragmentation. With segmentation, the chunks (also named segments) are stored as independent files, whereas in fragmentation the chunks (also named fragments) are stored within a single file. The streaming session consists of a sequence of HTTP-GET messages sent by the client to begin the download of consecutive sequence fragments (or segments). The herein presented analysis applies both to fragments and segments; in the following, without loss of generality, we refer to fragments.

Let us now consider a server storing $L$ versions of the same encoded video at different average rates $R_i,\ i = 1, \ldots, L$. Each sequence is encoded and parsed in fragments beginning with a random access frame. Each fragment corresponds to a fixed playback interval of $\tau_f$ s; the encoded fragment size in bits varies in a random way depending on the sequence activity and on the encoding parameters. We denote by $x_n$ the size (in bits) of the $n$-th sequence fragment. After an HTTP-GET, the fragment is downloaded in a time $\tau_n = x_n/r_n$, with $r_n$ representing the net bandwidth available to the application layer, namely the average TCP throughput measured during the fragment download time.

In HAS, the client does not request the $n+1$-th fragment unless the download of the $n$-th one is completed. Besides, in typical HTTP adaptive streaming systems, during an initial buffering stage the fragments are downloaded continuously until a predefined buffer level is achieved. Then, the client enters a steady state during which the fragments are played out at a pace of one fragment every $\tau_f$ s. During the steady state, the client waits for at least $\tau_f$ s before requesting the next fragment. This mechanism prevents buffer overflows, and avoids data pre-fetching mechanisms that are not suitable for the mobile streaming paradigm, where the client may randomly change the rate or even the selected video content on a temporal scale much shorter than that of Video on Demand streaming systems. Fig. 2 presents the timing of the server transmissions after HTTP GETs sent by the client.

Every time the actual download time $\tau_n$ exceeds the fragment playout time $\tau_f$, the buffer depletes accordingly, so that consecutive delays in the video data delivery may result in buffer underflow events and in rebuffering periods affecting the user QoE. In conventional HAS systems such events are prevented by reducing the rate of the selected encoded video. However, this results in several undesirable effects including visual quality fluctuations

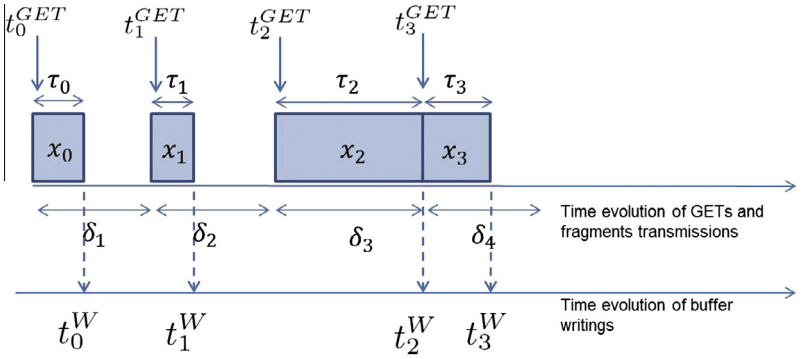



**Fig. 2.** Time evolution of the HTTP GETs sent by the client, the transmission of requested fragments and their writing in the playout buffer.

artifacts from a single user point of view [2] and oscillatory overall traffic behavior from a multiple user point of view. This is especially true in wireless ad hoc networks where challenging varying bandwidth conditions are often encountered.

In this context, we propose a sender-assisted HTTP streaming for ad hoc networks, where the sender (also called server in the following) providing the content can control the rate of the HTTP GETs of the clients in order to improve their QoE. For sender-assisted HTTP streaming, a Video Agent (VA) is activated at the server side, with the goal of improving the QoE of HAS across the wireless ad hoc network interconnecting the two users. The VA monitors the QoE of the client by comparing the amount of the actually streamed bytes and the amount that needed at the client for a given QoE.

A key advantage is that the VA, being at the server side, always has prior knowledge of the size in bytes of each and every encoded fragment of the video at a given quality. Therefore, once the client has selected one of the available $R_i$, the server knows the whole set $x_n$, for $n = 1, \ldots, k$, of fragments at the given rate. In this way, a cross-layer interaction between application and TCP layer at the VA side allows the VA to estimate the time required to transmit the $n$-th set of bytes $x_n$. Specifically, this can be computed by dividing $x_n$ by the estimated throughput, measured in terms of the Round Trip Time of the TCP protocol (that is a parameter known at the server side).

In what follows, we discuss how the interval between consecutive HTTP-GETs, denoted as $\delta$, can be set in accordance to a QoE oriented criterion, given the information available in the sender on the video fragments sizes and given an estimate of the net bandwidth. The scheduling is then communicated to the client, which consequently generates the actual HTTP-GET requests. We refer to this strategy as sender-assisted streaming.

## 4. Sender-assisted HAS analysis

We start by modeling the main system components that affect the optimal HTTP-GET scheduling problem. Specifically, we aim at identifying the set of $N$ intervals $\delta_n$ between consecutive HTTP-GETs that optimize a suitable QoE metric.

The selected QoE metric is based on the number of underflows and the duration of rebuffering events. With this objective, we now evaluate the buffer load at the receiver size.

We denote by $t_n^W$ the time instant at which the $n$-th fragment is fully downloaded in the receiver buffer, also known as the playout buffer (see the lower axis of Fig. 2). Besides, we denote by $t_n^R$ the time instant at which the $n$-th fragment is read for playback. Let us denote by $t_0^W = 0$ the instant of reception of the first fragment; the playback phase begins after an initial buffering delay of $\delta_B$ s, that is $t_0^R = \delta_B$.

Once playback starts, fragments are read at a regular rate; nonetheless, due to delay and jitter in fragment reception, caused by local fluctuations in both bandwidth and encoding rate, the buffer may underflow. In this case, playback is underflowed for a rebuffering interval $\delta_R$, during which fragments are loaded but not played out. After the rebuffering period, playout restarts at a rate $1/\tau_f$.

Hence, accurate modeling of the buffer behavior needs to take into account the occurrence of underflows. For this purpose, we introduce a binary sequence $s_n$ defined as follows: $s_n = 1$ when an underflow occurs corresponding to the $n$-th fragment and $s_n = 0$ otherwise. Then, the buffer evolution is characterized by the sequences $t_n^W,\ s_n,\ t_n^R$ defined as

$$\begin{cases} t_0^{GET} = 0 \\ t_0^W = t_0^{GET} + \tau_0 \\ s_0 = 0 \\ t_0^R = t_0^W + \delta_B \end{cases} \text{and} \quad \begin{cases} t_n^{GET} = t_{n-1}^{GET} + \max(\tau_{n-1}, \delta_n) \\ t_n^W = t_n^{GET} + \tau_n \\ s_n = 1/2 - 1/2\ \mathrm{sign}(t_{n-1}^R + \tau_f - t_n^W) \\ t_n^R = t_{n-1}^R + (1 - s_n)\tau_f + s_n\ \delta_R \end{cases} \tag{1}$$

for any $n \neq 0$.

We can now express the buffer occupancy based on the writing and reading instants in (1). Specifically, let us introduce the writing process $w(t)$ and the reading process $r(t)$ as

$$w(t) = \sum_n x_n u_{-1}(t - t_n^W),\ r(t) = \sum_n x_n u_{-1}(t - t_n^R) \tag{2}$$

The buffer occupancy (in bits) is then expressed as $w(t) - r(t)$. Besides, the number of underflows can be computed as $S = \sum_0^{N-1} s_n$. If we assume that each rebuffering period is deterministically set to $\delta_R$,[2] the overall re-buffering time $D$ is related to $S$ as $D = \delta_R S$.

A suitable QoE metric (to be minimized) for HTTP-GET scheduling is the number of underflows, which has direct implications in terms of the duration of rebuffering events. On the other hand, when operating on a set of consecutive fragments, there is a finite probability that the client can unpredictably end the streaming session. Therefore, we consider the following QoE-related objective,

$$\mathcal{C}(\delta_0, \delta_{N-1}) = \min_{\delta_n, n=0,\ldots N-1} \sum_0^{N-1} \rho_{SW}^n s_n \tag{3}$$

where $\rho_{SW} \in (0, 1]$ is a discount factor that takes into account the fact that the user can tear down the streaming session.

To avoid overflows at the receiver buffer and to limit the net throughput required by a single client to the sender, the download rate averaged on any window of $W$ fragments should not deviate significantly from the average video encoding rate $\bar{R}$ , i.e., $\sum_{k=0}^{W-1} x_{n+k}/(t_{n+W-1}^W - t_n^W) \leqslant \bar{R}(1 + \alpha)$, where $\alpha$ is a system design parameter.

To sum up, in the streaming session we recognize:

- an initial buffering stage, in which the fragments are downloaded consecutively without introducing unnecessary delay, and during which the buffer fills until a pre-defined level;
- a streaming stage, during which the fragments are downloaded forcing the intervals among HTTP-GETs; such intervals can be optimized so as to (i) minimize

[2] As an alternative, the rebuffering period can be set to the time required to download a certain number of data bytes, either fixed or varying so as to match a fixed video playout interval; such choices lead to a random characterization of the overall rebuffering time $D$.

the number of underflows occurring over all the session and (ii) bound the local average download rate to avoid buffer overflow.

To elaborate, first we observe that the optimization function $\mathcal{C}(\delta_0,\ldots,\delta_{N-1})$ is highly nonlinear with respect to the optimization variables $(\delta_0,\ldots,\delta_{N-1})$. Second, in the bandwidth varying conditions typical of a wireless ad hoc network, the overall session optimization is actually unfeasible. To optimally assign the intervals between consecutive HTTP-GETs, the actual download time intervals $\tau_n,\ n=0,\ldots,N-1$ need to be known. The fragment lengths $x_n,\ n=0,\ldots,N-1$ are indeed known at the server, and, if the actual throughput during the streaming session were known, the interval between the $N$ fragments could be jointly optimized. Due to the erratic throughput behavior, we limit the optimization to a window of $W$ fragments over which the prediction of the TCP throughput $r_n,\ n=0,\ldots,W-1$ is feasible. Therefore, we design a sub-optimal procedure described in the following section.

## 5. Sender-assisted buffer management

The sender-assisted buffer management procedure relies on restricting the scope of the prediction to a window of length $W<N$, which we refer to as the *look-ahead window*. Specifically, a window of $W$ consecutive fragments is considered. The fragment size sequence $x_n,\ n=0,\ldots,W-1$ is known at the sender. We initially assume that the relative channel throughput sequence $r_n,\ n=0,\ldots,W-1$ is also known; later, we show with the help of numerical results that the proposed approach when the throughput sequence is estimated.

On each window of length $W$, the sender evaluates the overall download time adopting the steady state client strategy, and compares it with the overall corresponding playout time, namely $\sum_{k=0}^{W-1}\max(\tau_k,\tau_f)\leqslant W\tau_f$ and signals the test result to the client. If the above condition is satisfied, the downloading proceeds as in the conventional case, namely $\delta_n$ is kept equal to $\tau_f$. On the contrary, if the overall download time exceeds $W\tau_f,\delta_n,\ n=0,\ldots,W-1$ are set equal to 0; this corresponds to a solution $\delta_0\leqslant\delta_0\leqslant\ldots\leqslant\delta_{W-1}$ matching the average video encoding rate. To maintain the steady state conditions, if $\sum_{k=0}^{W-1}\max(\tau_k,0)\leqslant W\tau_f$, the client waits for $W\tau_f-\sum_{k=0}^{W-1}\tau_k$ and then resumes download on the next window. Otherwise, if consecutive download the $W$ fragments has exceeded the expected window $W\tau_f$, the client immediately restarts download on the following window.

The proposed sender-based buffer management strategy is effective in improving the QoE at the client side. We show herein some examples from numerical simulations of a HAS session carried out using a video trace [13] encoded at an average encoding rate of 3.3 Mbps and streamed at a constant TCP throughput of suitably exceeding the average video encoding rate; a complete description of the simulation scenario is reported in Section 6.

Herein, we presents results of the analysis of different performance parameters, such as buffer occupancy vs. time, and the relevant QoE parameters, including the number of underflow events, and their duration.

The client requests the fragments corresponding to $\tau_f$ video seconds following to the selected (optimized or not) HTTP-GET scheduling strategy and it begins the playout phase after an initial delay $\delta_B$. The net bandwidth $r_n$ at the application layer, which is selected to be always larger than the net video rate, is kept constant throughout the session simulation. We show here the case of a client experiencing an average TCP throughput of 160% the average video encoding rate.

When an underflow event occurs, the client stops playout and continues downloading fragments, and then restarts for a rebuffering interval of duration $\delta_R$. Playout stops while the buffer is re-loaded with three fragments of encoded video, corresponding to a fixed playout time of 6 s. Due to intrinsic video bitrate fluctuations, the actual number of bits downloaded during rebuffering varies according to the variable fragment size in bits. The rebuffering interval $\delta_R$ varies as well.

In Fig. 3(a) we show the buffer time evolution at the client in the absence of a look-ahead-window. Specifically,

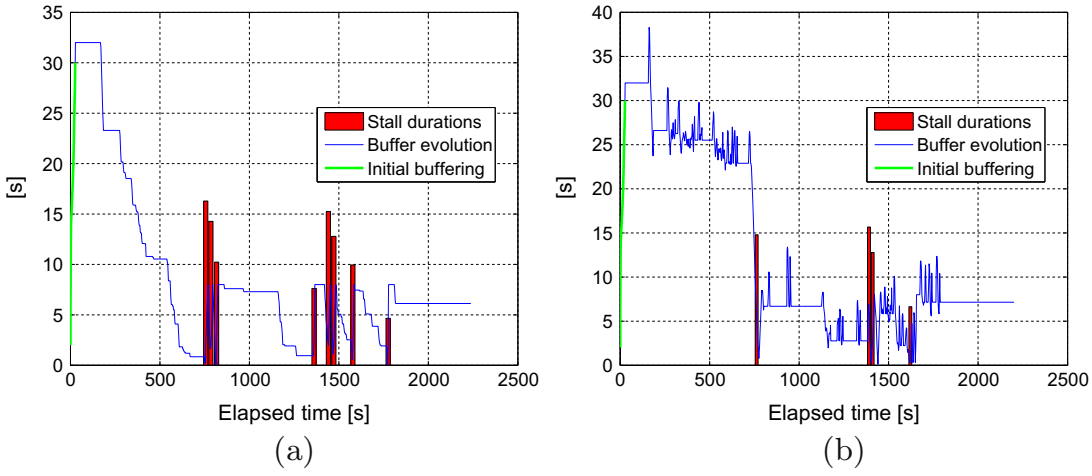



**Fig. 3.** Buffer occupancy measured in terms of video playout time versus time; underflow events highlighted; case of absence (a) and presence (b) of the look-ahead window.

the $y$-axis reports the buffer occupancy, expressed in video playout time, whereas the $x$-axis is the temporal axis. We recognize that in the first second of streaming, the curve has a very high slope corresponding to the initial buffering stage; next, despite the 60% extra bandwidth, the buffer begins depleting. This rapidly leads to a sequence of repeated underflow events. In Fig. 3(b), we represent the same performance metrics when considering a look-ahead window of $W = 6$. We recognize that, during the periods that would have caused buffer depletion, the look-ahead window mechanism is activated and the client downloads consecutive segments, alternating intervals at a locally higher throughput, to inactive intervals assuring the steady state condition (horizontal slope of the buffer occupancy). The number of underflow events drastically reduces, thus providing increased QoE.

To gain insight into the condition that determines the look-ahead window activation, we analyze the relationship between the fragment download duration and the fragment playout duration, namely $\max(\tau_k, \tau_f)/\tau_f$. After the initial buffering period, where each fragment is downloaded at the maximum available throughput, the steady state interval $\max(\tau_k, \tau_f)$ between consecutive HTTP-GETs should match the fragment duration $\tau_f$ (2 s in our simulations) and the ratio $\max(\tau_k, \tau_f)/\tau_f$ should equal one. When the fragment size $x_n$ is larger than the average size or the throughput $r_n$ is lower than expected, the interval between HTTP gets, forced to the maximum between the fragment duration $\tau$ and the actual fragment download duration, exceeds the fragment duration. In Fig. 4(a) we plot the value of $\max(\tau_k, \tau_f)/\tau_f$ vs. the fragment index $k$. We recognize that, even though the throughput is 6% larger than the average video encoding rate, a large number of fragments experience a download time larger than the corresponding playout time. This phenomenon causes rapid buffer depletion and yields a large number of underflow events. In Fig. 4(b), we represent the same ratio in the presence of the look-ahead window. We mark with negative spikes the indexes of the fragments on which the window successfully applies. Besides, we recognize that there are still a few fragments on which the ratio $\max(\tau_k, \tau_f)/\tau_f$ exceeds the unitary value. This occurs when, even though the look-ahead window is applied, the overall download time over the $W$ fragments still exceeds the playout time $W\tau_f$. Although residual delays are introduced during streaming, these are reduced in number with respect to the case when the look-ahead window is not used.

Based on this analysis, we deduce that the sender-assisted HAS buffer management procedure has the potential to increase the number of underflow events, which is indeed an important QoE metric in HAS. In the next section we analyze its application in wireless ad hoc networks.

**Table 1**
Video traces statistics.

| Trace | Bitrate avg. $\overline{R}$ (Mbps) | Frame size avg. ± std. dev. (kbit) |
|---|---|---|
| SpaceStation | 4.79 | 199 ± 9.7 |
| Titans | 3.79 | 158 ± 3.9 |
| Alice | 3.26 | 136 ± 1.9 |
| Monster | 1.57 | 114 ± 7.2 |

**Table 2**
IEEE 802.11 g parameters.

| | |
|---|---|
| Slot duration | 9 s |
| SIFS | 10 s |
| DIFS (SIFS + 2 * Time slot) | 28 s |
| CWmin | 15 |
| CWmax | 1023 |
| MAC header | 34 bytes |
| MAC maximum transfer unit | 2312 bytes |
| ACK | 14 bytes |

## 6. Performance evaluation

In this section, we evaluate the performance of the sender-assisted HAS buffer management in ad hoc networks through simulations. First, we observe that in ad hoc networks mobile devices experience large bandwidth fluctuations due to physical-layer fluctuations of the wireless channel; besides, user experiences random MAC delays

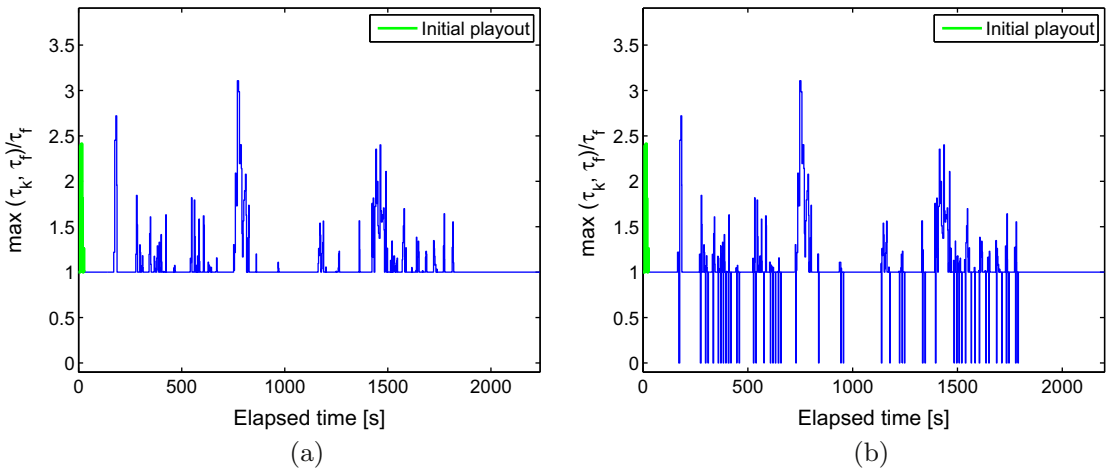



**Fig. 4.** Ratio between the steady state interval $\max(\tau_k, \tau_f)$ between consecutive HTTP-GET and the fragment duration $\tau_f$; case of (a) absence of the look-ahead window, (b) presence of the look-ahead window.

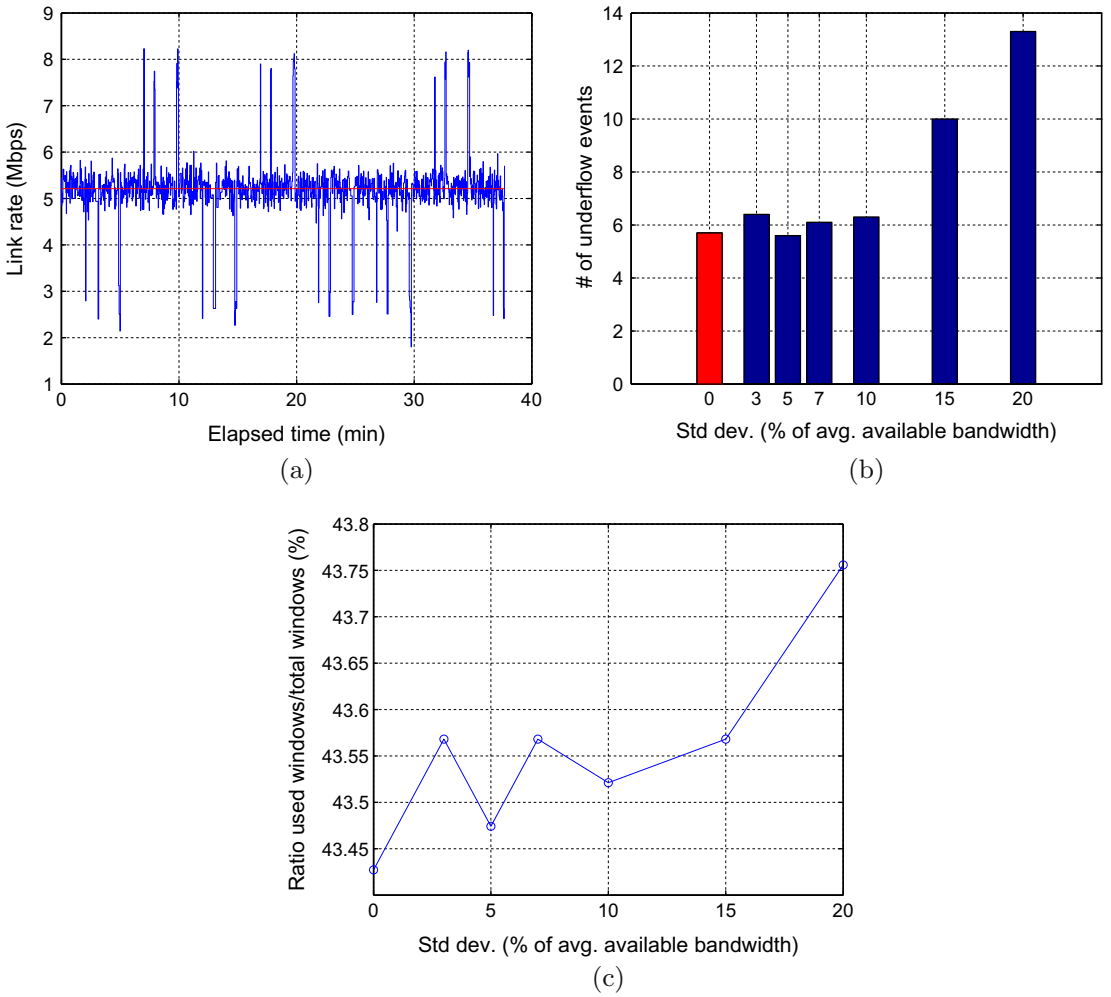



**Fig. 5.** Example of Wi-Fi bandwidth, number of underflow events and windows activation rate vs increasing Wi-Fi bandwidth fluctuations.

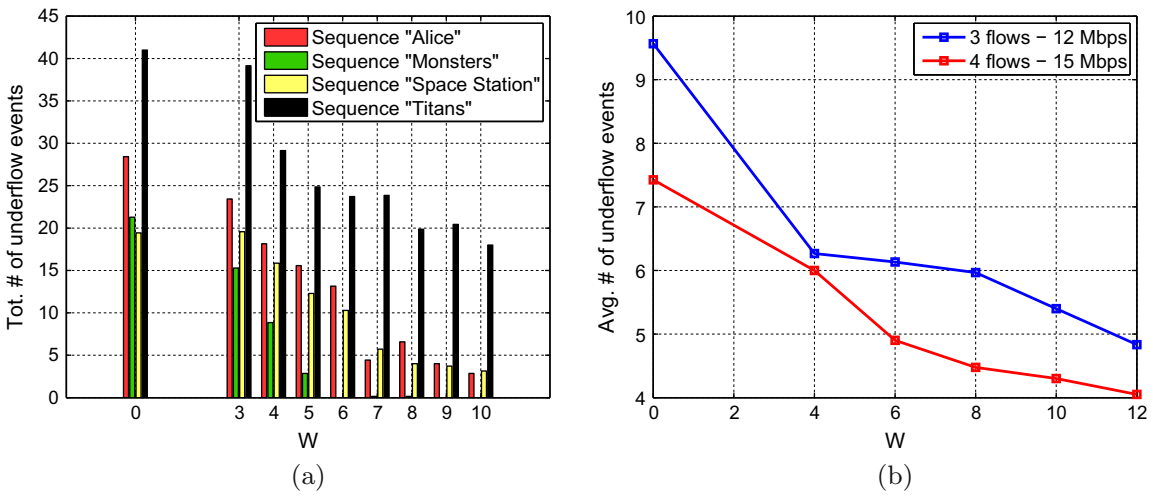



**Fig. 6.** Number of underflows as a function of the window size in presence of: (a) physical layer bandwidth fluctuations, (b) MAC collisions.

caused by the coexistence of multiple users. We show with the help of numerical simulation examples that the proposed sender assisted buffer management strategy provides an effective starting point to develop flexible HTTP video streaming strategies that can match specific challenges of ad hoc networks.

### 6.1. Simulation scenario

The simulations have been realized using Matlab in order to represent the overall HAS session running on top of different protocols. As far as the application layer is concerned, the source media used for representing the data streams are HD multi-view video files whose statistics and traces can be downloaded from the ASU video repository at the web page [13]. Specifically, as in [14,15], we employed a set of video traces from the database in [13]. The encoded video sequences SpaceStation, Titans, Alice, Monster, of 35.54 min each, have a spatial resolution of $1920 \times 1080$ pixels, a temporal resolution of 24 Hz, H.264 Multiview Video Coding encoded using a Group Of Picture (GOP) of 16 frames, with one B frame in between I/P key pictures, with a resulting average rate of 4.8, 3.8, 3.3, 1.6 Mbps. Each video presents intrinsic fluctuations of the frame size resulting in $x_n$ varying accordingly. Some characteristics of the adopted video traces are reported in Table 1.

The simulator tracks the evolution of the application layer buffer at the mobile station side; both filling and depletion in accordance to the interaction between the HTTP-GET requests and the net throughput offered by the lower layers. In order to assess the performance of the sender-assisted HAS strategy running on top of the CSMA/CA MAC protocol, the 802.11 g model has been implemented including a complete simulation of the CSMA/CA used at the MAC layer in order to allow a shared access to the medium. Table 2 reports the adopted IEEE 802.11 g parameters. In the following, we first show simulation results in presence of physical layer fluctuations, typically due only to the channel only (see Section 6.2); in this case, we consider only one active streaming session. Secondly, we show simulation results pertaining to the case of retransmissions caused by the collisions (see Section 6.3). Specifically, in this latter case we consider a variable number of concurrent point-to-point single-hop streaming sessions. We assume that the transmission range is greater than the maximum distance between the nodes; thus all nodes are within the same radio range. We also assume that the transmissions are ideal, i.e. there are no channel errors apart those due to collisions.

### 6.2. Streaming performance in presence of PHY layer fluctuations

In this section we describe simulation results typical of a wireless Wi-Fi scenario [2]. The herein adopted Wi-Fi bandwidth model reproduces the one in [2], but with the substantial difference that the frequency of the spikes is held, but the bandwidth between one spike and the other has been made variable and randomly changed from frame to frame. Fig. 5(a) plots an example of Wi-Fi bandwidth; in this case the average bandwidth (red straight line) is 6.08 Mbps. Both positive and negative spikes can be seen; finally the standard deviation of the variations is 5% of the mean bandwidth.

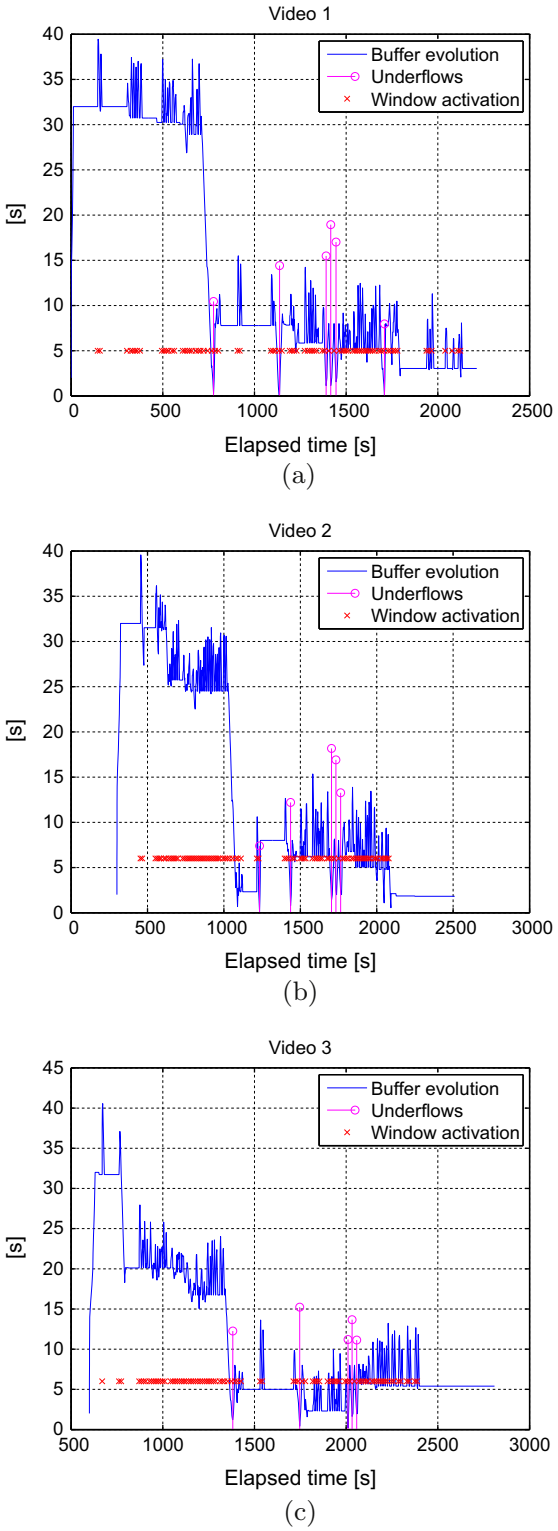


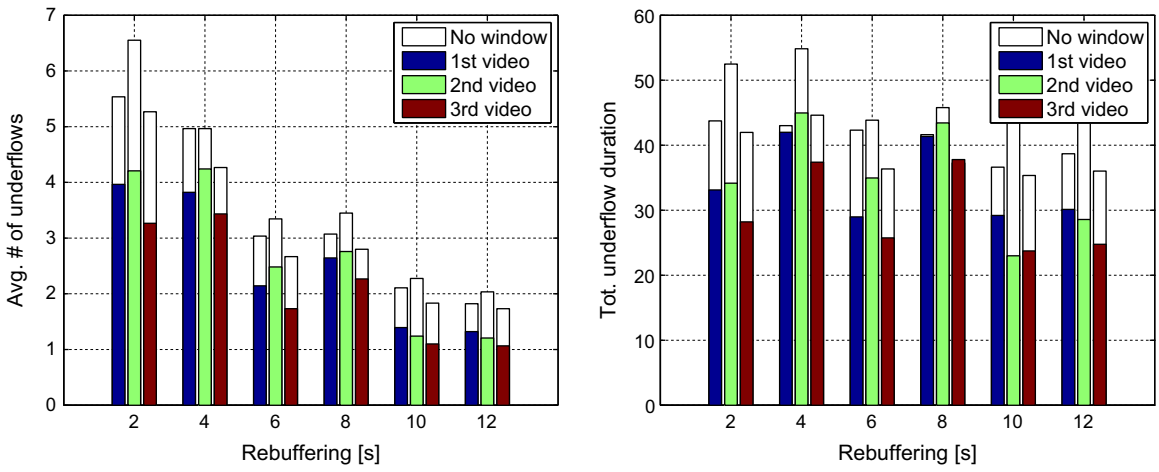

**Fig. 7.** Buffer evolution, window activations and actual underflows in case of videos starting in different time periods.

The following simulations have been carried out with the following setting parameters. The initial playout delay is 30 s, while the rebuffering interval duration is set to 6 s. We consider $W = 5$. Fig. 5(b) illustrates the QoE evaluated in terms of number of underflow events, averaged over 10 Montecarlo runs, vs. the standard deviation of the bandwidth fluctuations. As expected the number of underflows events increases with the variability of the bandwidth. The first bar in red is the QoE value for zero standard deviation, that is, when the bandwidth is constant. For the first standard deviation percentage values (0–10%) we intentionally considered denser measurements to assess the behavior of the system and evaluate the evolution of the look-ahed window. We show that the perceived quality is stable in this interval, since the look-ahead window allows changing the HTTP GET request frequency when the network conditions deteriorate. This effect can be seen also by observing the percentage of activations of the window (see Fig. 5(c)). When variations in the available rate increase, we observe a percentage increase in the number of times that the window is activated. This confirms the effectiveness of the proposed strategy in controlling the GET request frequency in reacting to throughput variations, however caused.

In Fig. 6(a) we compare the number of underflow events without look-ahead window ($W = 0$). Note that here we set the available bandwidth to 150% of the video encoding rate, and we consider a bandwidth fluctuating around its mean value with a standard deviation of 15 % of the mean. Besides, we assume that only the bandwidth mean is known at the sender while evaluating the window activation test. In Fig. 6(a), we appreciate that the look-ahead window strategy significantly improves QoE by reducing the number of underflow events. Finally, it is worth observing that other simulations, not reported here for the sake of conciseness, prove that the proposed algorithm works even when the throughput is estimated, provided that it is underestimated rather than overestimated. This means that the HAS sender assisted buffer management strategy can be effectively applied using the estimated throughput provided that the estimate is conservatively slightly biased.

### 6.3. Streaming performance in presence of MAC layer collisions

Finally, we analyze the performance of the look-ahead window mechanism in an environment where different senders experiment collisions. This is done by simulating simultaneous access to a standard IEEE 802.11 g network, with data rate variable between 8 and 24 Mbps and the MAC parameters reported in Table 2. In this scenario, we simulate different numbers of video streaming sessions, randomly initiated with a temporal displacement within 1–7 min concurrently accessing the network for contemporary HAS streaming. In Fig. 6(b), we plot the number of underflows, averaged over 10 Montecarlo runs, versus the look-ahead window length $W$. The case of absence of look-ahead window is represented as $W = 0$. We clearly appreciate the improvement in the number of underflows when the look-ahead window is used. In Fig. 7, we analyze three concurrent video streaming sessions of the video Alice, starting with 5 min shifts from one another. We report the buffer occupancy of the three video buffers, obtained for $W = 6$. We recognize the spikes due to the look-ahead window strategy and the steady buffer occupation resulting from its application. In this sense, collisions between streams do not affect the performance of the strategy. To assess the fairness among video streams, we consider concurrent streaming of three videos, referring to the same video sequence Alice but shifted one another by a random interval between 3 min and 7 min. We report in Fig. 8 the number of underflow events and the QoE metrics, averaged over 10 Montecarlo runs, for each video. We can see that the effect is fairly distributed on the three video sessions. Finally, to assess the performance of the look-ahead strategy in case of estimated bandwidth, in Table 3 we report the average number of underflows observed when the throughput is perfectly known and when it is inferred by a simple algorithm at the sender site




(a) Number of underflows. (b) Underflow duration

**Fig. 8.** Number of underflows and underflow duration in case of three videos randomly shifted: with and without the use of the window.

**Table 3**
Number of underflows for known or estimated throughput.

| Bandwidth (Mbit/s) | 8 | 10 | 12 | 14 |
|---|---|---|---|---|
| Number of underflows (Known throughput) | 12.1 | 5.1 | 1.8 | 0.5 |
| Number of underflows (Estimated throughput) | 12.3 | 5.3 | 1.9 | 0.6 |

from the number of experienced collisions. Specifically, each sender tracks the mean of the average number of collisions per fragment based on the ACK mechanism, and, based on an a priori established association between collision probability and number of simultaneously transmitting stations infers the mean data rate [bandwidth/estimated number of active flows] to be used in the window activation test. The performance is only slightly affected by the bandwidth being estimated rather than perfectly known.

## 7. Concluding remarks

In this paper, we presented a sender-assisted procedure that is able to improve the Quality of Experience (QoE) in HTTP Adaptive Streaming (HAS) services. The sender assisted HAS procedure exploits information on the encoded video content available at the sender side to derive the client originated encoded video fragments download requests. The procedure balances client-related quality issues, which would require intensive video chunk downloads to avoid playout underflows, with sender-related system constraints, which require the average download rate not to overcome the average video encoding rate. The proposed approach leverages knowledge of characteristics of the encoded video always available at the server side. Moreover, it paves the way to further joint optimization of concurrent streaming sessions issued by the same sender. Numerical simulation results show the significant QoE improvement achievable with the proposed sender-assisted buffer management procedure, especially in the challenging conditions that are often found in wireless ad hoc networks.

## References


[1] Cisco, Cisco Visual Networking Index: Forecast and Methodology, 2012-17 <http://www.cisco.com>.
[2] S. Akhshabi, A.C. Begen, C. Dovrolis, An experimental evaluation of rate-adaptation algorithms in adaptive streaming over HTTP, in: Proceedings of the Second Annual ACM Conference on Multimedia Systems, Ser. MMSys '11, 2011, pp. 157–168.
[3] S. Colonnese, P. Frossard, S. Rinauro, L. Rossi, G. Scarano, Joint source and sending rate modeling in adaptive video streaming, Signal Process.: Image Commun. 28 (5) (2013) 403–416.
[4] I. Sodagar, The MPEG-DASH standard for multimedia streaming over the internet, multimedia, IEEE 18 (4) (2011) 62–67.
[5] M. Gorius, Y. Shuai, T. Herfet, Dynamic media streaming over wireless and mobile ip networks, in: 2012 IEEE International Conference on Consumer Electronics – Berlin (ICCE-Berlin), 2012, pp. 158–162.
[6] S. Colonnese, F. Cuomo, T. Melodia, R. Guida, Cloud-assisted buffer management for HTTP-based mobile video streaming, in: 10th ACM symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, & Ubiquitous Networks (PE-WASUN '13), Barcelona, Spain, 2013.
[7] L. De Cicco, S. Mascolo, An experimental investigation of the akamai adaptive video streaming, in: Proceedings of the 6th International Conference on HCI in Work and Learning, Life and Leisure: Workgroup Human–Computer Interaction and Usability Engineering, Ser. USAB'10, 2010, pp. 447–464.
[8] S. Khan, Y. Peng, E. Steinbach, M. Sgroi, W. Kellerer, Application-driven cross-layer optimization for video streaming over wireless networks, Commun. Mag. IEEE 44 (1) (2006) 122–130.
[9] Y. Andreopoulos, N. Mastronarde, M. van der Schaar, Cross-layer optimized video streaming over wireless multihop mesh networks, IEEE J. Sel. Areas Commun. 24 (11) (2006) 2104–2115.
[10] M. Li, M. Claypool, R. Kinicki, Playout buffer and rate optimization for streaming over IEEE 802.11 wireless networks, ACM Trans. Multimedia Comput. Commun. Appl. 5 (3) (2009) 26:1–26:25.
[11] S. Colonnese, F. Cuomo, T. Melodia, An empirical model of multiview video coding efficiency for wireless multimedia sensor networks, IEEE Trans. Multimedia 15 (8) (2013) 1800–1814.
[12] G. Tian, Y. Liu, On adaptive http streaming to mobile devices, in 2013 20th International Packet Video Workshop (PV), December 2013, pp. 1–8.
[13] Video Traces. <http://trace.eas.asu.edu/videotraces2/3d/MultiviewJMVC/>.
[14] A. Pulipaka, P. Seeling, M. Reisslein, L. Karam, Traffic and statistical multiplexing characterization of 3-d video representation formats, IEEE Trans. Broadcast. 59 (2) (2013) 382–389.
[15] P. Seeling, M. Reisslein, Video transport evaluation with h.264 video traces, Commun. Surv. Tutor. IEEE 14 (4) (2012) 1142–1165.


**Stefania Colonnese** (M.S. 1993, Ph.D. 1997) was born in Rome, Italy. She received the Laurea degree in Electronic Engineering from the Universitá "La Sapienza", magna cum laude, Rome, 1993, and the Ph.D. degree in Electronic Engineering from the Universitá di Roma "Roma Tre" in 1997. She has been active in the MPEG-4 standardization activity on automatic Video Segmentation. In 2001, she joined the Universit'a "La Sapienza:, Rome, as Assistant Professor. Her current research interests lie in the areas of signal and image processing, multiview video communications processing and networking. She is currently associate editor of the Hindawi Journal on Digital Multimedia Broadcasting (2010). She served in the TPC of IEEE/Eurasip EUVIP 2011, (Paris, July 2011) and of Compimage 2012 (Rome, September 2012). She is Student Session Chair for IEEE/Eurasip EUVIP 2013. She has been Visiting Scholar at "The State University of New York at Buffalo" (2011), Erasmus Visiting teacher at Université Paris 13 (2012).

**Francesca Cuomo** received her Laurea degree in Electrical and Electronic Engineering in 1993, magna cum laude, from the University of Rome La Sapienza, Italy. She earned the Ph.D. degree in Information and Communications Engineering in 1998. She is Associate Professor in Telecommunication Networks at the University of Rome La Sapienza. Her main research interests focus on: Vehicular Networks, Wireless ad hoc and Sensor networks, Cognitive Radio Networks, Green networking. Prof. Cuomo has advised numerous master students in computer science, and has been the advisor of 8 Ph.D. students. She has authored over 80 peerreviewed papers published in prominent international journals and conferences. She is in editorial board of the Elsevier Ad-Hoc Networks and she has served on technical program committees and as reviewer in several international conferences and journals. She served as Technical Program Committee Co-Chair for the ACM PE-WASUN 2011 and 2012 ACM International Workshop on Performance Evaluation of WirelessAd Hoc, Sensor, and Ubiquitous Networks. She is IEEE Senior Member.

**Raffaele Guida** obtained his Bachelor's degree from the University of Sannio (Italy) and Master's degree from the University of Rome "La Sapienza", both in Telecommunication Engineering. Since 2014 he is working as an ICT Engineer at Aenduo in Rome. His research interests include multimedia network, video streaming technologies and mobile cloud computing.

**Tommaso Melodia** is an Associate Professor with the Department of Electrical Engineering at the State University of New York (SUNY) at Buffalo, where he directs the Wireless Networks and Embedded Systems Laboratory. He received his Ph.D. in Electrical and Computer Engineering from the Georgia Institute of Technology in 2007. He had previously received his "Laurea" (integrated B.S. and M.S.) and Doctorate degrees in Telecommunications Engineering from the University of Rome "La Sapienza", Rome, Italy, in 2001 and 2005, respectively. He is a recipient of the National Science Foundation CAREER award, and coauthored a paper that was recognized as the Fast Breaking Paper in the field of Computer Science for February 2009 by Thomson ISI Essential Science Indicators and a paper that received an Elsevier Top Cited Paper Award. He is the Technical Program Committee Vice Chair for IEEE Globecom 2013 and the Technical Program Committee Vice Chair for Information Systems for IEEE INFOCOM 2013, and serves in the Editorial Boards of IEEE Transactions on Mobile Computing, IEEE Transactions on Wireless Communications, and Computer Networks (Elsevier), among others. His current research interests are in modeling, optimization, and experimental evaluation of wireless networks, with applications to cognitive and cooperative networking, ultrasonic intra-body area networks, multimedia sensor networks, and underwater networks.