An Interactive Video Streaming Architecture Featuring Bitrate Adaptation

Ivan Alen Fernandez, Christophe De Vleeschouwer
ICTeam, Université Catholique de Louvain, Belgium
George Toma, Laurent Schumacher
FUNDP Namur, Belgium
Email: {ivan.alen,christophe.devleeschouwer}@uclouvain.be
{george.toma,laurent.schumacher}@fundp.ac.be

***Abstract*— This paper describes an interactive and adaptive streaming architecture that exploits temporal concatenation of H.264/AVC video bit-streams to dynamically adapt to both user commands and network conditions. The architecture has been designed to improve the viewing experience when accessing video content through individual and potentially bandwidth constrained connections. On the one hand, the user commands typically gives the client the opportunity to select interactively a preferred version among the multiple video clips that are made available to render the scene, e.g. using different view angles, or zoomed-in and slow-motion factors. On the other hand, the adaptation to the network bandwidth ensures effective management of the client buffer, which appears to be fundamental to reduce the client-server interaction latency, while maximizing video quality and preventing buffer underflow. In addition to user interaction and network adaptation, the deployment of fully autonomous infrastructures for interactive content distribution also requires the development of automatic versioning methods. Hence, the paper also surveys a number of approaches proposed for this purpose in surveillance and sport event contexts. Both objective metrics and subjective experiments are exploited to assess our system.**

***Index Terms*— interactive streaming, clip versioning, RoI extraction, bitrate adaption, H.264/AVC.**

I. Introduction

Streaming services are becoming the highlight of value-added mobile services. Lately, the number of streaming applications developed on smart and cell phones increased dramatically, to give access to more and more multimedia contents. Based on the latest developments of the wireless data network, and the adoption of compression technologies such as H.264 [1]–[3], several media players have been designed and implemented for mobile handsets. In addition, due to the massive diversification of mobile users, and because of the shortage of mobile network bandwidth, the concept of client profile has been earning more and more importance. Its default purpose is to offer different streaming quality levels and different contents to the clients, depending on the purchased services.

This paper introduces an integrated architecture to support service diversification through adaptive and interactive streaming capabilities. The proposed system aims at offering personalized experience when accessing high resolution video content through individual and potentially bandwidth constrained connections. Fundamentally, the underlying architecture relies on the concatenation of pre-encoded clips to adapt to a pre-defined set of user commands, as well as to the network conditions. On the one hand, the user commands typically give the client the opportunity to select interactively a preferred option among the multiple video clips that are made available to render a given scene, e.g using different view angles, or different zoomed-in and slow-motion factors. On the other hand, adaptation to the network bandwidth is obtained through intelligent and dynamic switching between the multiple versions of the content that have been generated by encoding the same video at different quality (and thus bitrate) levels. The implementation of an effective switching strategy adapts the bit-rate of the delivered stream to the available bandwidth of the current link. It ensures accurate control of the client buffer, which is fundamental to reduce the client-server interaction latency while maximizing video quality and preventing buffer underflow.

Now that we have introduced the main principles of our proposed architecture, we detail the motivations underlying our investigations, and stress the arguments that make our work original, relevant and timely.

The need for interactive mobile streaming solutions naturally arose from the two following observations. At first, due to mobile network bandwidth limitation, it is often not possible to transmit large rate video streams, which in turns constraints the resolution of the streamed video images. On the other hand, content produced for conventional wired broadcast or captured by surveillance networks is gaining in resolution. As a consequence, this content has to be down-sampled to be accessed through mobile networks, thereby losing a lot of its value. A possible solution might be to manually produce a second version of the content that is dedicated to low resolution accesses. However, this solution is expensive and not ap-

This paper is based on "Browsing Sport Content Through an Interactive H.264 Streaming Session," by I. A. Fernandez, F. Chen, F. Lavigne, X. Desurmont, and C. De Vleeschouwer, which appeared in the Proceedings of the 2nd International Conference on Advances in Multimedia (MMEDIA), Athens, Greece, June 2010. © 2010 IEEE.

This work was supported in part by Walloon Region projects Sportic, Walcomo and Belgian NSF.

Manuscript received April 15, 2011; revised July 15, 2011; accepted October 15, 2011.

doi:10.4304/jcm.7.4.265-280

propriate in many application scenarios (e.g. surveillance or real-time post-production of broadcast content). For those cases, the only alternative is to design automatic video processing systems that produce low resolution content out of the native high-resolution content. Simple down-sampling of the native content is not appropriate since it results in the loss of many visual details. A preferred solution consists in cropping the initial content, to focus on Regions-of-interest (RoI). Such kind of automated tools have already been investigated in the literature [4]–[9], and a general conclusion is that none of the existing method is 100% reliable in the way it defines RoIs. Therefore, human supervision of the process is always required to check that the automatic content adaptation system behaves properly. Besides, in some cases, more than one region are likely to be of interest for the user. Our interactive framework proposes to circumvent those issues by allowing the end-user to decide about the rendering option he/she would like to visualize among a finite set of options that have been pre-computed based on automatic systems. Conversely, the above observation also reveals that the recent advances in automatic analysis and production of content [10]–[13] offer an unprecedented opportunity to deploy interactive and personalized services at low cost. In particular, the ability to identify the spatial regions or the temporal actions of interest in a video directly supports the automatic creation of several options to render an event, e.g. by skipping non-interesting actions or zooming on RoIs. Hence, no manual pre-processing of the content is required any more before actual exploitation of the interactive infrastructure.

Our paper develops and assesses the integrated components involved in the deployment of an interactive and adaptive streaming solution. The main contributions of the paper include:

- The design of the adaptive streaming architecture, based on the temporal concatenation of pre-encoded video clips. In practice, client-transparent switching between versions is enabled by splitting each produced video in short clips that are temporally pipelined by the server, based on user's requests, network conditions estimation or video content metadata and interaction strategies. From the client's perspective, everything happens as if a single pre-encoded video was streamed by the server. This is in contrast with the solution developed in [14], which supports continuous interactive Pan/Tilt/Zoom navigation while streaming high-resolution content, but therefore relies on dedicated spatial-random-access-enabled video coding.
- The development of control mechanisms, to adapt the streaming rate to network bandwidth. A number of works have already addressed the problem of adapting the sending rate of a streaming session to match the observed network conditions. Our work fundamentally differs from those earlier contributions by the fact that it puts a strong emphasis on maintaining a small client buffer all along the streaming session, thereby reducing the interaction latency[1]. This is obtained through the definition of an original and cautious probing strategy combined with careful analysis of the RTCP feedbacks.
- The definition of interactive commands, and the development of automatic methods to extract multiple rendering options out of a single high-resolution video. Such automatic versioning methods are indeed required to support the deployment of fully autonomous infrastructures for interactive content distribution. To address this issue, we survey some of our earlier contributions [15], [16] to explain how different video streams can be extracted out of high resolution content in a fully automatic manner both in the videosurveillance [17]–[20] and sport broadcast context [21]–[24]. Spatial and temporal adaptations are considered. Spatially, we crop the high resolution content to extract a lower resolution image that focuses on some automatically detected RoI(s). This solution provides an alternative to the regular sub-sampling of the initial content. Temporally, the automatic segmentation of the event into semantically meaningful actions or events can support fluent and efficient browsing across the video sources. Notably, a significant advantage of the interactive access scenario, compared to the fully automatic creation of personalized content, is that it gives the last decision about the way to vision the content in the hands of the final user. This is especially important since most video analysis tools remain prone to errors. Subjective tests have been considered to assess the experience offered to end-users by the proposed interactive architecture. They demonstrate the relevance of the approach.

The remaining of the paper is organized as follows: Section II presents the proposed architecture for interactive video streaming, through client-transparent temporal concatenation of pre-encoded video clips. In Section III, we describe the algorithm for bit-rate adaptation. Section IV introduces the interaction commands, together with automatic tools to version video surveillance and broadcast content. Finally, Section V presents some qualitative and quantitative results to validate our system. Section VI concludes.

II. Interactive Browsing Architecture

The main objective of our architecture is to offer interactivity to any client of a mobile video streaming session using an H.264/AVC compliant player. At the same time, the architecture supports bit-rate adaptation, so as to match dynamic bandwidth constraints while maximizing playback quality. Both capabilities are offered based on a

[1]Reducing latency by trashing the client buffer when the user sends a clip switching command is not a desired solution since it would result in a waste of resources. It would also significantly increase the system complexity, due to the need to inform the client about the actual fraction of the buffer that should be trashed to save latency while preserving transparent and continuous streaming.

generic content pipelining feature. As depicted in Figure 1 the communication is established with the client through the Real Time Streaming Protocol (RTSP). Video is forwarded using the RTP protocol. RTCP feedbacks are then used for dynamic bit-rate adaptation, while RTSP commands interpretation supports interactive browsing capabilities. In this section, we briefly introduce the different modules involved in the architecture. Additional details regarding bitrate adaptation and content versioning for interactive streaming will be presented in Sections III and IV, respectively.

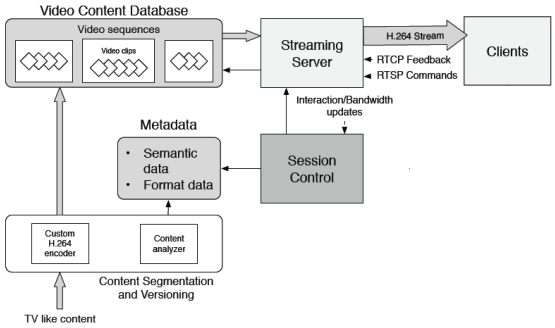

Figure 1. Diagram of the architecture's workflow

A. Architecture of the Streaming Server

The architecture on the server side is composed of 3 main components: the content segmentation and versioning unit, the streaming server and the session control module.

1) *The Enhanced Content Creation Unit* fills the Video Content Database, without actively taking part afterwards in the streaming system. Its purpose is threefold:

- It analyses the TV like video content to identify RoIs and produce several versions (replay, quality, view angle etc.) and zoomed-in alternatives of the content.
- It divides the video sequences in small pieces that are encoded based on H.264 according to the requirements explained in sections II-B and IV.
- It generates the metadata (shown in Section II-C) that are required to model and define the interactive playing options and quality versions associated to the different clips. Therefore, the metadata information is used by the session control to monitor the streaming session in response to the interactive user requests.

2) *The Streaming Server Module* is the core of the system, which supports client-transparent interactive streaming through on-the-fly content pipelining. Client-transparent temporal content pipelining allows the server to stream multiple successive video streams in a single session, without negotiating with the client the establishment of a new streaming session. Hence, with this feature the server is able to change the streamed content while maintaining a unique output stream and keeping the existing session uninterrupted. As a result, both a temporal and computational gain are achieved as the client does not need to establish more than one streaming session. The streaming server delivers all the data content through the Real-time Transport Protocol (RTP).

3) *The Session Control Module* determines, at every moment, which video clip has to be transmitted next. This unit consequently decides the video clips that are concatenated based on requests from the client, the estimated network status and on alternative versions offered by the enhanced content creation unit. Therefore, the session control is an essential part of the system, as it monitors almost any information flowing through the system.

B. Temporal Content Pipelining

Temporal content pipelining is the technique that allows a streaming server to juxtapose multiple streams in a single continuous sequence, so that multiple streams can be forwarded to the client through a unique and fluent streaming session. The key for implementing this functionality is the session control module using the advanced features of the H.264 codec [25], regarding the encoding parameters transmission.

The H.264 standard defines two kinds of parameter sets: sequence parameter sets (*SPS*) and picture parameter sets (*PPS*). The first applies to a wide range of frames, while the latter only applies to specific ones. Every Network Adaptation Layer (*NAL*) unit containing data information includes in its header a parameter linking it to a *PPS*, which in turn links to a specific *SPS*. In our architecture, all clips are encoded independently from each other. Since the first *NAL* unit of an H.264 segment always contains the *SPS* and the *PPS*, multiple sequences can be transmitted consecutively without any interruption, and the output is still compliant to the H.264 standard. When necessary, on the client's side, a unique sequence is received, which however, is built step by step by the server. The *SPS* are updated between two pipelined segments.

C. Session Control and Metadata

The session control processes the user's feedback, the RTCP statistics, and uses the metadata associated to the clips, to decide which clip should be delivered next. As described in Section IV, the metadata information is generated by the content (segmentation) and versioning unit, and is stored within a Extensible Markup Language (XML) file.

From a semantic point of view, we distinguish two different cases on the server side, depending on whether storytelling continuity has to be ensured or not when switching between clips. When temporal continuity is required, clip switching can only occur at the intersection between two consecutive clips. Those instants are depicted with vertical dashed lines in Figure 2. For this reason, the sequences have to be divided in very small clips, as each clip has to be completely delivered before switching. Otherwise the browsing capabilities would

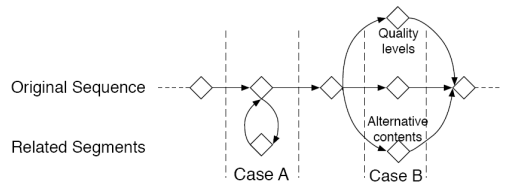

Figure 2. Metadata considered structures

only be offered on a coarse granularity basis. In cases for which temporal continuity is not required, as happens when the user wants to skip some non-relevant content, any buffered data in the server is discarded, so as to start reading the new clip file as soon as possible, thereby reducing to the minimum the overall latency associated to the switching mechanism. Like in the previous cases, the playback proceeds without causing any decoding error and the streaming behaviour is not damaged, performing the switching flawlessly.

From a functional point of view, two different kinds of temporal relationships between clips are envisioned, as depicted in Figure 2. Case *A* typically corresponds to an optional inclusion of a replay within the stream. The sequence playback is resumed after the additional content without any offset. The same relationship can be considered if *target advertising* is inserted in the stream according to the user preferences. In contrast, case *B* considers contending versions, which means that only one version is actually included in the output stream. As an example, possible contending alternatives include videos at different resolutions (zooming), fast-forward/regular speed mode, and different video quality versions. Hence, this case is extensively exploited to react to the interaction commands sent by the client. In Section IV, we define those commands, and survey a number of solutions that can be used to automatically generate the multiple rendering options that are of interest to the user, when visualizing high-resolution surveillance or sport event content. In addition, this case *B* also provides the possibility to switch between different quality (and thus bit-rate) levels, depending on the bandwidth limitation and in a completely transparent way for the user. In Section III, we explain in details how network probing can be implemented to infer the state of the network by increasing the burstiness of sent-out packets. We also describe how RTCP feedback monitoring can be exploited to decide at which rate the content should be forwarded by the server. Those two aspects are fundamental to adapt to mobile network fluctuations, thereby preserving video quality while limiting the size of the client buffer, and thus the interaction latency.

D. Interactivity with Video Player

The system's interactivity relies on the RTSP commands that are exchanged between the server and the client. This communication channel is already established and can be used to obtain feedback from the client. The user must be able to send a switching command, which induces a system response according to its content. The browsing features are then triggered by sending the appropriate request to the server.

A standard RTSP message is used by the client player to communicate its feedback. The considered RTSP command in our architecture is *OPTIONS*, as described in [26]. Combined with the header *Require*, it provides an efficient and simple way to signal user's feedback during the transmission. A specific value in the field of this header such as "*Switching-feature*", directly associates the RTSP command with the browsing capabilities of our server. A new line in the header, starting like "*Switching-Parameter:* " signals and conveys the different possible requests of the user (zooming, replay or fast forward mode). The mentioned interactive requests are associated one-by-one to new-functional buttons of the player's interface. These buttons consequently trigger a RTSP command from the user side when they are pressed. As an alternative, many clients such as the VLC Video Player, implement a seek function by sending the command PLAY with a parameter called Range [26]. Not only does it trigger a stream playback, but it may also seek inside the stream. While our server has been designed to attend such request, the browsing capabilities are further limited by this scenario. As an example, in a multi-angle camera scenario, the user has to send several requests to switch in between all the available sequences in round-trip without being able to access directly to the desired one.

III. Automatic Bit-Rate Adaptation Through Version Switching

To ensure a good user experience when streaming in wireless networks, it is necessary to adapt the streaming rate to frequent bandwidth variations. The video bit-rate should be reduced in the presence of congestion or a low quality link, but should be kept as high as possible to maximise the image quality. This section investigates the control problem in the particular case of our proposed interactive streaming framework. In Section III-A we present previous work related to stream adaptation and motivate why a new technique is needed to improve interaction delay, besides received video quality. Section III-B shows how congestion or signal degradation is detected. Eventually, Section III-C introduces our proposed rate adaptation algorithm, which prevents client buffer starvation in presence of congestion (to preserve playback fluency), while keeping the streaming rate close to the available bandwidth. Sections III-D and III-E explain the proposed probing mechanism used to determine the available bandwith and to keep a reasonably small buffer to preserve interactivity.

A. Motivation of the Chosen Adaptation Algorithm

In an interactive streaming scenario, there are two elements that contribute to improve the user's experience:

- The received video quality;
- The reactivity of the streaming system to user interactions.

Maximising the received video quality is a challenging task, especially in the context of varying mobile network conditions. This means that in the case of bandwidth constrained connections the playback should remain smooth without re-buffering or jerkiness and when the network allows, the viewer should receive the best achievable image quality. This translates into the ability to select the appropriate encoding rate of the chosen content, based on the available throughput.

A number of bit-rate adaptation techniques have already been proposed in the literature, but they generally don't care about interactivity. Even more, some of these solutions like the ones presented in [27], [28] and [29] require a custom-built client which would limit the use of the adaptive streaming framework to those specific media-players.

We propose a bit-rate adaptation algorithm that attempt to maximize both the received video quality and the system reactivity. In an interactive system, it is essential to keep the reaction time as low as possible, the delay between a request at the client side and the consequence of that action in terms of played content should be minimised. This delay has 3 contributions:

1) The server side delay, if the request arrives just after the initial frames of a clip (temporal consistency is targeted).
2) The end-to-end (E2E) delay, from the server to the client through the network
3) The time required to empty the pre-roll buffer, since there is no remote possibility to flush the video buffer of a player.

The first contribution depends on how the video stream is split into clips to support interactive services. As explained in Section II-C, we recommend to use short clips, thereby reducing the upper bound of this delay to 700 ms. More details about this issue can be found in [15]. The second component is imposed by network at hand and is independent of the server and the client. The third component of delay depends on the client buffer fullness when an interaction command is launched by the user. It can be reduced by trying to keep the client buffer level as low as possible. This can only be achieved in the presence of fine rate adaptation mechanisms. Those mechanisms have a double objective: they attempt to maximize the streaming rate while preventing the buffer to become empty when the network conditions become worse. Working with a small buffer makes the problem especially challenging since it increases the risk of interrupting the playback. Hence, rate adaptation is severely constrained by interactivity, which imposes to keep small buffers.For this reason, in Section III-C we propose an original probing mechanism that empties the buffer to a certain level by a pause in the transmission, the so-called gap, while the following burst of packets brings back the buffer to its normal position. This approach allows to probe the network because, during the burst, data is forwarded at a faster instantaneous rate than the average streaming rate. Both the congestion detection and the network probing methods are further described below.

B. Estimating Network State

Network conditions are estimated from the information sent back by the client through the periodic RTCP reports. Specifically, the Receiver Reports (RR) and the Sender Reports (SR) from the RTCP protocol will be used to compute the RTT, jitter, packet loss and average throughput.

The RTT[2] can be computed by the server using the method presented in [30]. A fast increase in the RTT suggests that congestion is about to take place in one or more links across the network path. Because the variation nature of the instantaneous RTT is spiky, two variables will be used to characterize its evolution: a smoothed RTT and the RTT deviation. The formulas are inspired by the method adopted to compute TCP Retransmission timer [31], but have been modified so as to:

- Know whether the deviation increases or decreases
- Increase the impact of the instantaneous measurements compared to past reports

The formulas write as follows:

$$\begin{aligned} &\text{Smoothed}_{RTT} \\ &\quad = \quad (1-\alpha) * \text{Smoothed}_{RTT} \\ &\qquad\quad +\alpha * \text{Instant}_{RTT} \end{aligned} \tag{1}$$

$$\begin{aligned} &\text{Deviation} \\ &\quad = \quad (1-\beta) * \text{Deviation} \\ &\qquad\quad +\beta * (\text{Instant}_{RTT} - \text{Smoothed}_{RTT}) \end{aligned} \tag{2}$$

where α and β are both 0.5.

The network state is then inferred from the evolution of the Deviation parameter over a specific number of consecutive RTCP reports. As a rule of thumb, we consider that a network encounters congestion once the Deviation value is higher than 100 ms for two consecutive reports. The rest of the paragraph illustrates the empirical study that has led to this rule. Figures 3 and 4, plot the formulas in (1) and (2), along with the instantaneous RTT in two distinct scenarios. In Figure 3, the Network Emulator (NetEm) Linux module is used to reduce the network bandwidth to the video bitrate, for a limited time period. In Figure 4, the RTT distribution is based on measurements in live 3G networks [32]–[34]. We focus on GPRS measurements as they exhibit the largest RTT variations. One observes that the Deviation only goes above 60-70ms (in absolute value) when the current transmission rate is close to the maximum available bandwidth, but remains under this value in absence of congestion even in the case of a GPRS connection, whereas the jitter is higher than in other mobile networks. Also, the authors of [35] have reported that in 90% of the cases, the jitter was smaller than 100ms in their measurements.

Consequently, if the absolute Deviation value is higher than 100 ms for two consecutive reports, this should be

[2]Despite being a two-way time measurement, the RTT is regarded as a good estimate of the E2E delay.

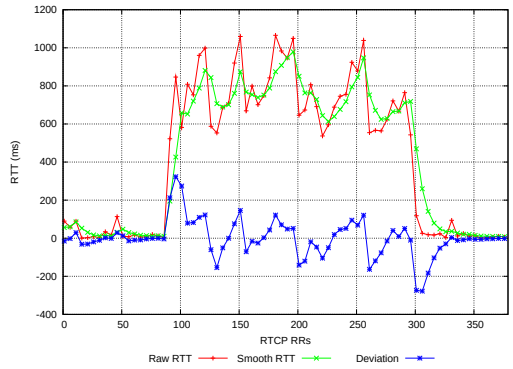

Figure 3. RTT evolution in a congested network. Bandwidth reduction applied at 90s and removed at 300s.

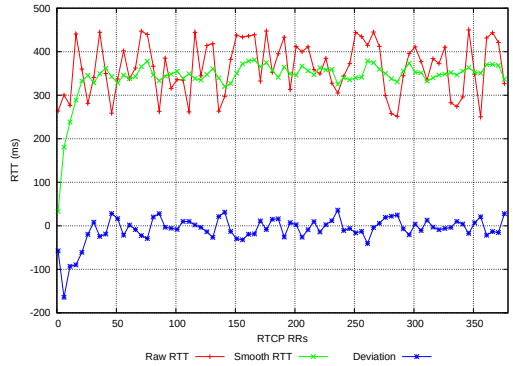

Figure 4. RTT evolution in GPRS network.

interpreted as a sign of congestion. To increase decision robustness, a large number of RTCP RRs should be taken into consideration. However, when the media client supports a standard implementation of the RTP/RTCP protocol, it sends RTCP reports every 5 s (like VLC, QuickTime). Waiting for more than 2 reports would therefore lead to a reaction time longer than 10 s, which is not acceptable. Hence, in practice, decision about congestion state is taken based on two observations of large RTT deviation.

As explained above, alternative clues for congestion detection lie in the fields of the receiver report that are related to lost packets, namely the fraction loss and the cumulative loss. The first represents the ratio between the number of lost packets and expected number of packets since the emission of the last RR or SR, while the latter represents the number of lost packets since the beginning of the session.

A combination of the two reports will be used to decide about congestion and to consider a down-switch in the transmission rate. Specifically, using the current and previous RRs, the server can compute the total number of lost packets for the reporting interval:

$$\begin{aligned} \text{nr\_lost\_packets} &= \text{current\_cumulative\_report} \\ &\quad - \text{previous\_cumulative\_report} \quad (3) \end{aligned}$$

This value, combined with the fraction loss provides insight into the loss status. Congestion is inferred when a sufficient number of packets has been lost on a sufficient long history, or equivalently when a sufficient number of packets have been lost on a sufficient recent history. In practice, congestion is assumed when packet loss ratio is higher than 10% and the total number of lost packets between the current and the previous RTCP report is higher than 10 packets. These threshold parameters were chosen after several simulations under a QDisc limited Ethernet network and in a real WiMax and WiFi access networks. Also, the results presented in [35] and [32]–[34] were taken into consideration

C. Adaptation Algorithm

As stated in Section II-C, the *Streaming Server Module* has the ability to switch between different H.264 encoded clips, meaning that it can seamlessly switch between versions of the same video encoded at different rates. We therefore define a down-switch as the change in the streaming chain to a lower quality encoding and the up-switch the change to a higher quality encoding. The adaptation algorithm is based on congestion detection and network probing mechanism. Our proposed scheme is presented in Figure 5 as a Finite State Machine (FSM) with three main phases: Initialisation, Probing and Normal.

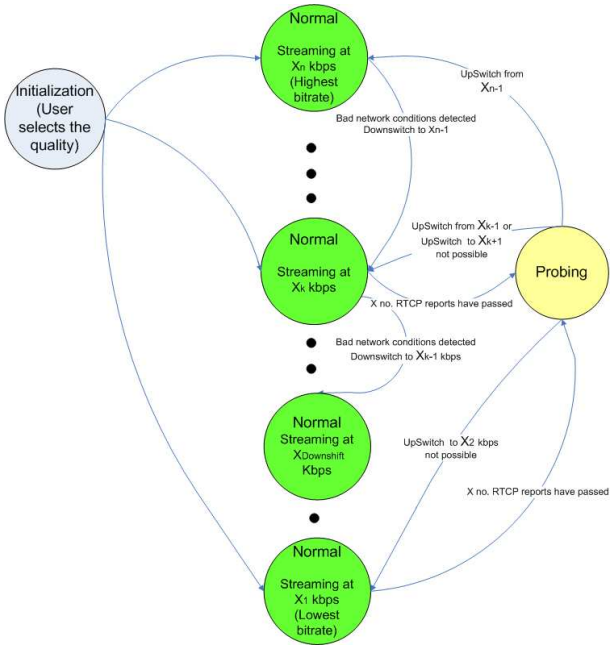

Figure 5. Adaptation algorithm

1) Initialisation state: This is the initial phase of the algorithm, it includes the RTSP negotiation and network discovery, when the server collects statistics about the current state of the network. The first two RTCP reports are used for the initialisation of $Smoothed_{RTT}$ and $Deviation_{RTT}$. In this phase, the server sends the video encoded at a bitrate that is close to the quality requested by the user.

2) Normal X_k state: In this state the server sends the media at a constant rate of X_k kbps and analyses the RTCP reports, where $k \in \{1..N\}$ and N is the number of supported bitrate versions. From here, depending on

network conditions, the server can remain in the same state, can pass to X_{k-1} through a down-switch or can go to the $Probing$ state to assess whether there is enough bandwidth to up-switch to X_{k+1}. Consequently, it is necessary to define possible bitrates for the X_k states and the moments when a change of state is needed.

According to [27], a Fast bitrate Decrease Slow bitrate restore Up (FDSU) approach is the most suitable way to assess network congestion. Using this method, the server switches to a clip encoded at approx. 60% of the current encoding, avoiding severe image degradation. However, a slow bitrate increase implies producing many quality versions of the same content, which puts a burden on the post processing and storage of several versions. Hence, we will use a Fast bitrate Decrease Fast bitrate restore Up (FDFU) approach. This approach implies that the number of X_k states can be reduced to 3. For example in a cellular environment, each state would be defined to encompass the average rate of a cellular generation, e.g. EDGE (140 kbps), UMTS (200 kbps) and HSPA (350 kbps), as measured in [36].

A down-switch is performed when the network bandwidth cannot support the current streaming rate. As explained in Section III-B, this means that is should be triggered when the RTT Deviation is higher than 100ms for 2 consecutive RTCP reports, or when packet loss ratio is higher than 10% and the total number of lost packets between current and previous RTCP reports is higher than 10 packets. To increase the responsiveness of the algorithm, if the deviation exceeds 300 ms, the server will immediately down-switch to a lower rate because such high values indicate severe network congestion. The server repeats the down-switch only if the deviation continues to increase.

After a configurable number of RTCP reports, the server will go into the probing state only if the network does not have signs of congestion.

3) Probing state: This is an auxiliary state, in which silent gaps and bursts of RTP packets alternate in order to estimate whether additional bandwidth is available in the network. The main idea behind this technique is to send the video frames at a higher rate (burst) to put the network under stress. If the bandwidth limit is close to the current bitrate, the packets sent at a higher rate would queue in the network buffers and the RTCP would report high RTTs at the server. Consequently, from the RTT values reported in the RTCP reports, the streaming server can assess whether the available network bandwidth is high enough to switch to a higher bitrate. If this is the case, then the server should switch from X_k kbps to X_{k+1} kbps. Otherwise it will resume regular streaming at X_k kbps.

The advantage of this probing technique compared to the tools that compute the available network bandwidth by sending packet trains or packet chirps (for instance abing [37], pathchirp [38] or Wbest [39]) is that it does not send extra data over the network. In addition, there is no need for deploying a tool on the client side to analyse the packets.

A possible drawback of the proposed approach is the fact that the amount of data which can be sent at a faster rate is limited due to the risk of client buffer overflow. To overcome this issue, the burst of RTP packets has to be followed or preceded by a pause in the transmission, the so-called gap). This allows the data from the buffer to be consumed, or to refill the buffer to its average occupancy respectively. If we choose to have the burst first, followed by the gap, we minimize the risk of emptying the client buffer, but increase the average size of the buffer during the probing process which impairs interactivity.

Since we aim to reduce the interaction time as much as possible, we did choose to pause the transmission first and then send the burst to probe the network.

D. Choosing burst and gap size for probing period

When probing the network, we would like to know if the current available bandwidth allows to switch to the next encoding rate, which should be at about 60% higher than the current rate, according to FDFU approach. However, since streaming closely to the bandwidth limit could lead to high RTT and packet loss, we have decided to up-switch only when the bandwidth limit is almost twice as high as the current streaming rate.

Because we want to have a neutral impact on the buffer after a complete probing cycle, the length of the gap is strictly related to the length of the burst. We therefore define the Gap, in seconds, as below:

$$\begin{aligned} \text{Gap}_{duration} \\ = \quad & (Burst_{Length}) * \frac{1}{FPS} - (Burst_{Length} - 1) \\ & * \frac{1}{FPS} * \frac{1}{Probing_{Factor}} \end{aligned} \tag{4}$$

where $Burst_{Length}$ represents the probing duration expressed in number of frames, FPS is the video frame-rate and the $Probing_{Factor}$ represents the frame-rate increase. For example, for a 25fps video, if we stream at twice the rate (the $Probing_{Factor}$ would be 2, streaming at 50 fps, but keeping the same presentation time, so the frames would be displayed at the correct speed to the viewer) for 31 frames, the 32nd frame would represent the Gap of 660 ms. We have discovered however that sending the video at twice the frame-rate, does not put a significant load on the network because the burst period is short compared to the Gap. The $Probing_{Factor}$ was increased to 4, with the same $Burst_{Length}$ of 32 frames which returned a Gap of 970 ms. We could not increase the $Probing_{Factor}$ further because the Gap would increase even more and the media player buffers would need a higher playout value which would affect interactivity. Moreover, for most conventional players, the maximum gap we can make before emptying the buffer is about 1s for robust transmission. As depicted in Figure 6, to stress even more the network, the probing cycle is repeated 6 times which covers the period of 2-3 RTCP reports.

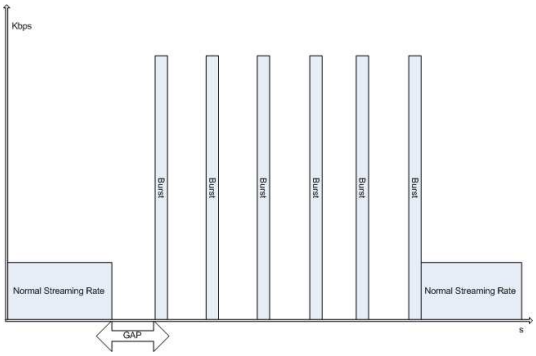

Figure 6. Probing cycle

E. Definition of up-switch thresholds

After each probing cycle, the server has to decide whether to up-switch or not, based on the RTT deviation observed from the RTCP reports. As derived in Annex A, one can associate the expected available bandwidth to the observed deviation in RTT. As explained in Section III-D we aim to switch up only when the available bandwidth is twice as high as the current encoding rate. The mathematical derivations in Annex A, when parametrized based on actual network measurements, reveal that if the deviation observed after probing is smaller than 100ms, there is a 90% chance that the available bandwidth is equal or higher than twice the rate. Hence, we have adopted a threshold of 100ms in RTT deviation to decide whether to up-switch (deviation below threshold) or not (deviation above threshold).

IV. Automatic content definition and versioning

In previous sections, we have presented a an adaptive streaming framework that gives the user the opportunity to switch interactively between multiple versions of a visual content. However, in addition to user interaction and network adaptation, the deployment of fully autonomous infrastructures for interactive content distribution also requires the development of automatic versioning methods. Hence, this section completes the picture by introducing a number of approaches proposed for this purpose in two different scenarios: sport event broadcasting and video surveillance. Typically, the (automatically) produced low-resolution versions include a sub-sampled version of the native video, plus one or several zoomed-in (and cropped) versions, each one focusing on one (of the) RoI(s) detected in the native high-resolution video. In terms of interactive functionalities, users can select the original video which offers a general view of the scene or select videos that focus on specific RoIs. In some application scenarios, replays of hot spots actions are also proposed to the user.

A. Interactive commands and browsing options

In the soccer video context, three browsing capabilities are offered: alternative fast forward mode, replay of hotspots and zooming over the RoI. Figure 7 presents the interaction strategy supported by our framework, initially introduced in [15].

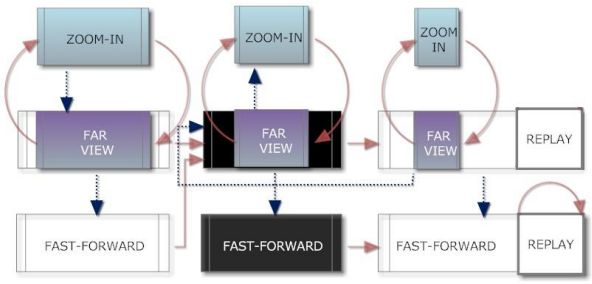

Figure 7. Switching control strategy. Dashed arrows represent potential requests from the user, while continuous arrows depict automatic connections between versions based on the interaction strategy. The central segment corresponds to an important action of the match.

Fast forward mode is available for the user during all the playback. When this mode is active, the video replay of the involved actions is skipped. Every time the playback reaches a highlight segment of the game, the fast-forward mode is automatically switched to regular mode catching the attention of the user. Zoom-in is available in regular mode for far camera shots. The viewer has always the faculty to decide the mode that he/she considers convenient to receive. At the beginning of every new segment the user can request the replay of the segment that has been displayed previously. After the repeated segment is displayed, the playback of the current segment where the replay was requested is recovered without any offset.

For video surveillance, automatic RoI extraction methods are used in order to extract the moving objects of the scene. Examples of such methods are presented in [16], [40] and [41]. Alternative videos are then generated by cropping the areas of the image containing the objects of interest. An example is depicted in Figure 8. The last column contains the available video versions at a given time instant.

Figure 8. New "zoomed versions" of video stream. First row is the original video stream. Second row is created when a first mobile object appears in the scene. Third row is created when a second object is detected and tracked (the abandoned luggage). Forth row is the stream that includes the two mobile objects.

B. Temporal consistency and division in shots, clips and segments

To provide the temporal browsing capabilities, different levels of division granularity are considered. Starting from the native raw content, our system automatically splits it into non-overlapping and semantically consistent segments. Each segment is then divided into shots, based on conventional view boundary detection. Shots are finally split in small clips. These clips support our browsing capabilities during the whole playback in a temporally consistent way, following the metadata considerations described in II-C. Hence, switching between versions should be allowed between shots, meaning that a boundary between shots should also define a boundary between clips. The same holds for segments.

In the surveillance context, the shot denotes the entire video sequence, and segments are segmented based on activity detection. In the sport broadcast context, a shot is defined as a portion of the native video that has been produced with constant or smoothly evolving camera parameters. This approach is based on average difference between consecutive histogram-features as already described in [15]. Afterwards, the shots are classified in different view types: replays, non-grass close-up views and close, medium or far grass view camera. At the end, far views are computed in order to obtain an alternative zoomed-in version that is stored in the enhanced content creation unit. Interested readers may refer to [15] for more details about shot definition, and view type classification. They can also refer to [16] for the automatic generation of zoomed-in versions in case of far view.

Figure 9 presents an example of our framework applied to soccer game. The resolution of a football game video extracted from TV broadcasting is automatically adapted to a small device screen. The zoomed-in sequences are offered to the user as an alternative replacing the original segments upon request.

(a) (b)

Figure 9. Original and processed zoom versions of the same frame.

Finally, segments are defined as shots closely related in terms of story semantic, e.g., shots for an attacking action in football. Proposed by the authors in [42], semantically meaningful segmentation is achieved based on a general diagram of state transition, which consists in one round of offense/ defence as described in Figure 10. For completeness, we note that audio or video analysis tools [43] have been designed to highlight key actions automatically, thereby offering additional browsing granularity. We conclude that many algorithms do already exist to fed or interactive framework in a fully automatic manner, making its practical deployment realistic, since manual intervention is not required to create dedicated interactive content.

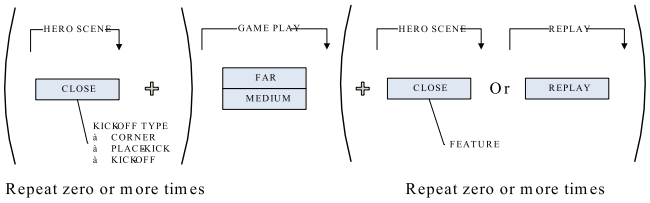

Figure 10. General structure of a gameplay and view types.

V. TESTS AND RESULTS

A. Performance evaluation of the proposed platform

In this section we perform 3 types of tests to show the improvements in interactivity delay and in quality of experience over the same solution without rate adaptation. Although scalability tests have not been made, being a VoD platform, it inherits the typical VoD scalability issues where multicast and broadcast techniques are not used.

1) Convergence speed of the Rate Adaptation algorithm: In the first experiment we test the reactivity of the algorithm, with the parameters presented in section III-C. The set-up consists 2 PCs, one hosting the modified version of a LiveMedia555 streaming server and the other hosting VLC media player, connected via 100Mbps Ethernet interfaces. The available bandwidth between the two can be reduced using the Linux Traffic Control tool, to simulate congestion or signal degradation in wireless networks. During several streaming sessions the available bandwidth was reduced close to, or below the current streaming rate and the reaction time between bandwidth reduction and an actual down-switch was registered. The cumulative distribution function for the delay is plotted in Figure 11, where 3 cases can be distinguished:

- when the bandwidth drops to a value closer to the current streaming rate, the down-switch delay is in average equal to 10s, which represents the duration of approximatively 2 RTCP reports.
- when the bandwidth drops to a value at least 20% lower than the current streaming rate, the system reacts faster, in about 5-6s, which represents the duration of about 1 RTCP report.
- the bandwidth drops during probing, the down-switch delay is approximatively 10s as well, the duration of 2 RTCP reports

Table I summarizes the performance of the adaptation algorithm in case of a decrease in the available bandwidth.

Bandwidth drop level	Bandwidth = current rate	Bandwidth < current rate	Bandwidth drops while probing
Avgerage reaction time	11.4s (2 RTCP RR)	6.4s (1 RTCP RR)	10.5s (2RTCP RR)

TABLE I.
SUMMARY OF DOWN-SWITCH REACTION TIME

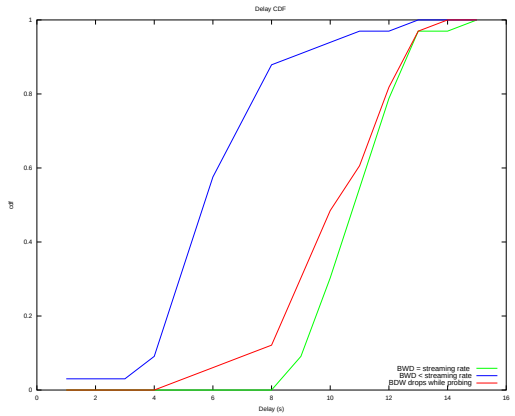

Figure 11. CDF for the down-switch delay

The reaction time directly depends on the frequency of the RTCP reports, and the results obtained in this paper present the worse case scenario, where the media player used sends 1 RTCP report every 5s.

The up-switch delay depends on the probing frequency, which was fixed to each 6 RTCP reports for the duration of the tests, and on the probing success. So if the bandwidth allows it, the system would choose a higher streaming rate after 8 RTCP (probing frequency + probing duration) reports, the equivalent of approx 40s. If frequent feedback is used, for example 1 RTCP RR each second, the reaction time would be reduced to 8s. Probing success percentage is given in Table II. We can observe that even in the ideal case when the bandwidth is almost twice as high (195%) as the current rate, we have an up-switch success of only 71%. This means that in 30% of the cases the quality should have been increased by the server, but it was not. The reason is that the video rate is not perfectly constant and it might happen that during probing the actual bandwidth limit may be smaller and therefore higher deviation may result. In the case of 150% and 165% the success rate is 50% which can be considered as a false positive because we only want to increase the quality when available bandwidth is twice as high. This event is not a harmful one though because the network should be able to support the higher rate for a while and in a moving scenario the signal strength will continue to increase so no harm was done in the end. The most important is the low percentage of up-switching in the worst case scenario (120% limit) when the server increases the video quality introducing congestion in the network.

Bandwidth limit (QDisc)	120%	150%	165%	195%
Up-switch success rate	16%	49%	51%	71%

TABLE II.
UP-SWITCH SUCCESS RATE

Compared to the adaptive streaming solution proposed in [44], although we did not have access to their platform to test it in similar conditions, we can see that performance is similar when detecting a bandwidth drop (approx. 6s delay) but going back to the original quality takes longer in our solution. This may be influenced by the frequency of the RTCP reports, which is not specified in [44]. The adaptation algorithm is implemented to achieve a trade off between fast reactivity and stability and without imposing special restrictions to the media player. During down-switch, by design, the system reacts faster, because increased RTT reflects a problem and we want to avoid high delays and packet loss. The up-switch takes longer because frequent switching in video quality is not desired.

2) Tests in a WiMax Access Network: Because the platform is intended to be used with clients connected in a wireless environment, two test scenarios were prepared, first with the client connected in a WiMax access Network and second with the client connected to a WiFi router. The WiMax setup is presented in Figure 12.

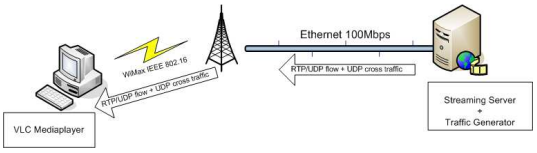

Figure 12. WiMax setup

For this test, where available 3 different versions of the content on the server: first encoded at 2.5Mbps, second at 1.7Mbps and the lowest quality encoded at 800Kbps. Since the capacity of our WiMax channel is about 6.5Mbps, we simulated a drop in the available bandwidth by sending additional UDP cross traffic over the air interface at a rate of 4.5Mbps. The duration of the bandwidth limitation is set to about 25s, similar to the throughput variations observed in [45] at driving speed.

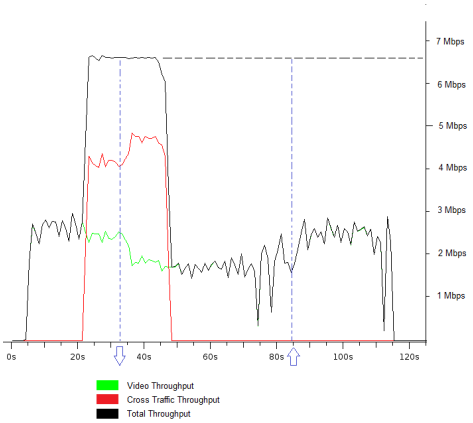

Figure 13. Throughput evolution during WiMax test

In Figure 13 when the cross traffic is sent (red line in the figure), the total throughput (the black line) reaches the maximum capacity of 6.5Mbps, which is not enough to send the whole 7Mbps of data (2.5Mbps video + 4.5Mbps cross traffic). The server decides to switch to the next lower encoding quality after approx 10s and will up-switch back in another 40s, after the bandwidth limitation has passed. If we compare the RTT evolution to the case

where no rate adaptation was used in Fig. 14, we can see that the maximum value of RTT is lower and also the congestion period is minimised. In this way interactivity speed is not affected suffered and also the packet loss rate is kept to 0 so the QoE was maximised. In Table III, we have the average of the RTT during the congestion period obtained from 4 different runs of the same experiment.

Experiment type	Average RTT during cross traffic
With Adaptation	270ms
Without Adaptation	650ms

TABLE III.
AVERAGE RTT DURING CONGESTION

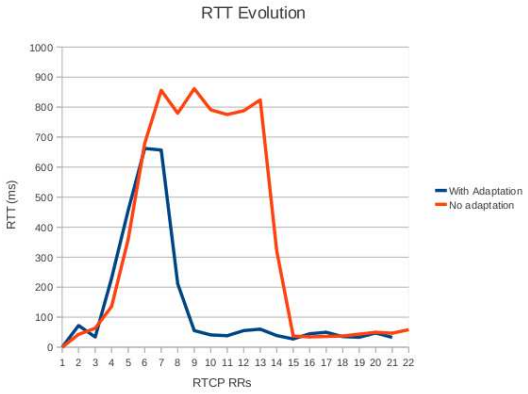

Figure 14. RTT evolution in the WiMax test

3) Tests in a WiFi Access Network: In the WiFi test, the client was connected to a WiFi 802.11g router configured in NAT mode with port forwarding enabled to allow UDP traffic. Although a multi hop experiment has not been considered in this paper, experiments performed in [46] show that delay evolution is similar to the one observed in single hop paths. The streaming session started near the access point, then the PC was moved away about 20m from the router loosing line-of-sight and then returned to the initial position. During this mobility test, the signal was not lost, but suffered degradation so the available bandwidth decreased with distance and increased back again when the client approached the WiFi router. In this case, the content versions available on the server were encoded at 380Kbps, 240Kbps and 180Kbps. Again, the experiment was ran once with the rate adaptation enabled and once without adaptation with the results shown in figures Fig. 15 and Fig. 16

We can see again that the RTT was kept low to improve the interactivity delay and that packet loss was also limited by the switch to a lower encoding, more suited to the network conditions. Compared to the WiMax and Qdisc tests, packet loss was more severe in the WiFi environment and it affected the image quality even if the available bandwidth was higher than the streaming rate.

In Table IV, we have the average packet loss during the whole streaming session obtained from 5 different runs of the same experiment.

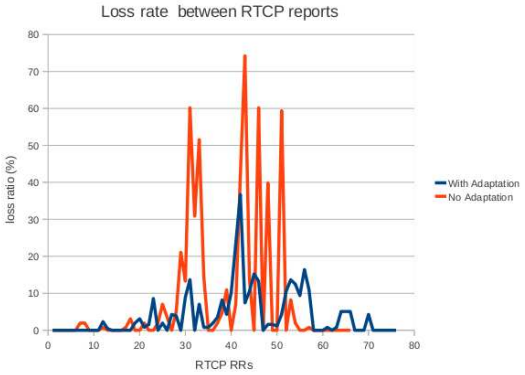

Figure 15. Loss rate during WiFi test

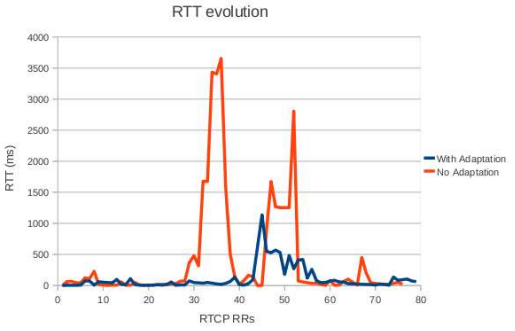

Figure 16. RTT evolution during WiFi test

Experiment type	Average packet loss
With Adaptation	3.6%
Without Adaptation	8.2%

TABLE IV.
AVERAGE LOSS RATE FOR THE WHOLE STREAMING SESSION

B. Cost of Compression Induced by Segmentation, and Switching Latency

The streaming abilities are implemented using the liveMedia library that has been extended to deliver H.264 files. Our tests have revealed that the fact that the video sequence is segmented in small clips, as described in Section II, does not penalize the fluency of the streaming playback. On the server side, although clips have to be pipelined dynamically in the transmission buffer, the processing load is not dramatically increased, and the correct rhythm of delivery of RTP packets is preserved even during the probing stage.

However, slight bitrate cost and some constraints are applied over the encoder H.264, in order to enable adaptive streaming and video content segmentation:

1) The overall compression speed is clearly damaged as the encoding process of every sequence is divided in the multiple clips and several alternative versions are provided. Nevertheless, the scenarios we consider are based on *on demand* video content. Hence, all the clips are preprocessed and included in the video database *in advance*, and because of this, the performance is not damaged.

2) Every new clip has to start with a new Instantaneous Decoding Refresh (*IDR*) frame, penalizing the encoding flexibility. Therefore, the segmentation in multiple pieces

of every sequence constraints the maximum size of the *GOP* (Group of Pictures) to the size of the encoded clips. Moreover, bitrate overhead is resulting from the use of *IDR refresh* frames. For this reason, a trade-off between the time of the system's response to the user's feedback, and the size of the clip has to be achieved, as every clip has to be completely delivered before starting to send the new one (due to the constraint of switching between versions in a temporally consistent way). If the clips are short, the system switches the playback very fast independently of the instant when the user's request is received. However, the penalty in terms of bitrate increases when the clip size decreases (*GOP* is also small increasing the bitrate). The opposite result occurs if the clips are longer. In our simulations we used sequences encoded at 25 *fps* and clip segmentation approximately every 15 frames. On the one hand, using 1 *GOP* per clip, a *GOP* of 15 frames is good enough in order not to penalize the global bitrate. The global loss in quality in PSNR in the luminance and chroma is less than 0.5 dBs respect to encoding the same sequence without the *GOP* constraint across several bitrate configurations (as depicted in Figure 17). On the other hand, the maximum latency in the server due to the clip segmentation is less than 700 milliseconds, as in the worst of the cases, the server has just sent the first frame of a new clip when receiving the request to switch the content. This delay is a good approach as depending on the Round Trip Time (*RTT*) of the wireless network and the pre-roll buffer of the player, the minimal delay is already in the order of 2 seconds. This cost is also low when measuring the quality loss with other techniques such as Structural similarity (SSIM). In this case, when handling very low bitrates (150-600 kbps) the loss can drop until 0,002 meanwhile for higher bitrates (1200-2000 kbps) this difference is lower than 0,0005.

3) Finally, it is also important to consider the increment of bitrate due to the *SPS* and *PPS* headers that are used in every new video clip. In the case that all the video sequence is encoded once, they have to be sent to the client just one time at the beginning. This is not the case when the sequence is split in several clips as in our framework. In Table V we include the increment of bitrate for different video resolutions at different levels of quality (by modifying the quantization parameter: *QP*). As we can observe, the cost of the headers is very low and almost negligible for higher quality encoding parameters (*QP*=16). The size of the header is almost constant in every case, independently of the encoding parameters that are being used. Hence, when the quality of the image is increased at the cost of spending bitrate, the related cost of the headers gets lower and lower. The video segmentation occurs again approximately every 15 frames.

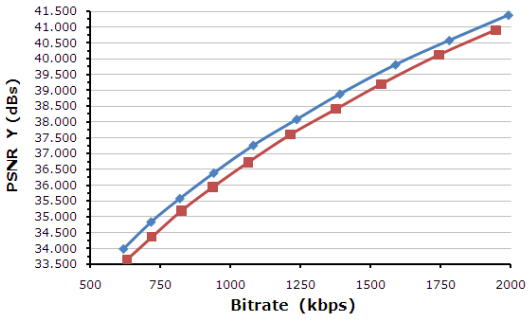

Figure 17. Video quality comparison in the luminance component when applying or not the GOP constraint. Red line represents a sequence encoded with GOP 15, while the blue line depicts the same sequence encoded without GOP restrictions. The Bitrate is computed for different QPs.

Sequence dimensions	**Quantization Parameter**	**Bitrate increment** (%)
176x144	16	0,86
176x144	32	5,95
352x288	16	0,68
352x288	32	5,73
720x576	16	0,76
720x576	32	3,84

TABLE V.
INCREMENT OF BITRATE USING VIDEO SEGMENTATION DUE TO THE REQUIRED SPS AND PPS HEADERS TO SYNCHRONIZE THE DECODER

The global interaction delay has also been measured through several tests (100 samples per case). This delay is considered as the difference between the time the user presses de request button and the new content starts to be displayed in the player. Hence, it sums up the time needed to forward the client request to the server, the time elapsed before the server gets the opportunity to switch between clips (this is proportional to the clip duration), and the buffer size (we assume no buffer flushing). As shown in Table VI the global delay depends on the probing strategy, and is decreased thanks to the proposed adaptive streaming strategy. Pausing the delivery of content before a new probing attempt increases the margin of time the server has to switch to another clip, due to a client request. Obviously, during the pause, one should take care not to empty the pre-roll buffer of the client, which is regulated from the beginning of the video transmission. In contrast, if the probing is implemented by increasing the delivery rate before pausing the system, the interaction delay is increased compared to a system without probing (see last line of Table VI).

Experiment type	Average delay
Without Adaptation	2.28 s
With our model	2.18 s
With burst before pause	2.44 s

TABLE VI.
AVERAGE GLOBAL DELAY TO THE USER REQUEST.

C. Validation of the Interactive Features Through Subjective Tests

Our platform was tested through questionnaires answered by 20 different people. Te viewers were asked to exploit the interactive features of our system in different video sequences related to sport content and video surveillance respectively. The soccer demo contained 10 minutes of a match with some highlights of a match. The

video surveillance sequence, of similar duration, consisted on scenes in open air parking where different people and vehicles pass by. From the results of the experiments, we depict the satisfaction of the viewers with our browsing capabilities and the way they handle them when they get used to the platform. The latest was approached by a second round of demos after filling the questionnaire.

In soccer, the three browsing features were valued by at least 90% of the viewers as very or quite profitable (5 or 4 out of 5 in our score ranking). The interaction strategy was also generally approved. Some users might still prefer the non zoomed-in version proposed by default for far view shots or the resumption of a segment after a *replay on demand* from its beginning. The transparent switching from fast forward to regular mode at the beginning of a highlight was well appreciated for all. The favourite feature was the replay (65% of the users) while zoom-in(out) was the most used according to our records of. The main complaint of the users was that the zooming factor, although well centered over the RoI, had only one level and should be more aggressive to be distinguished from the original version. Nevertheless, this issue is associated to computer vision algorithms and not to the proposed practical functionalities.

In video surveillance, all the users considered very or quite profitable the capacity of selecting single RoIs from the general view and focusing over them. Also the users do not perceive any loss of quality when dealing with HD sequences where the view is split in 4 different cameras and they can focus the one with an available RoI. The round-trip strategy is clear for all but 80% consider it not practical when dealing with many RoIs at the same time due to the limitations of the GUI. Most of the viewers also appreciate the video contents based on focusing over two or more RoIs (85%) and the original view alternative in which the detected RoI is compressed with higher quality than the background (95%).

In global, all the testers considered the video streaming fluent enough compared to other standard streaming servers. No one could notice any issue devoted to the change of rate delivery due to the bitrate adaptation algorithm as the video rate did not change or got anyhow stuck. Temporally consistency was also generally approved and well appreciated. 40% of the users still noticed some small video artefacts in some occasions after pressing a button for an request. This factor just related to the video player performance was still not considered as damaging (not ranked in any case more than 3 out of 5 in our scale). The interaction delay was considered a very important factor for 85% of the viewers and particularly critical in video surveillance. Finally, 70% of the users considered that the video player interface could be slightly improved. Although considered simple and handy, for a 55% of the users the GUI should be more intuitive. More buttons or plots over the video should then be used for a more direct, easier and clearer navigation over the different content alternatives.

VI. Conclusion

In this paper, we described a flexible interactive streaming system, over one underlying key mechanism: temporal content pipelining, which allows to switch the video content at whichever point of the playback in a temporally consistent way. This mechanism uses the client's feedback, requiring only one open streaming session per user and no advanced implementation mechanisms. Furthermore, we implement a streaming quality adaptation algorithm based on the RTCP feedback received from the client. In this algorithm, rather than just focusing on its general purpose, a novel probing technique embedded to decrease the interaction delay of our interactive system. Experimental results validate our adaptive rate algorithm and show that the video segmentation does not have any effect in the fluency of the streaming playback and in addition, the bitrate is not significantly increased. Therefore, the browsing features do not damage the global performance of the system. We also present three different switching capabilities when streaming video soccer content: zooming over RoIs, fast forward and additional replays selection. All together, subjectively increases the perceived quality of the streaming experience. The profits of our architecture mainly rely on supporting personalized content selection according to the interaction with the viewer and the automatic video quality adaptation. Finally, our framework is also able to include, for example, targeted advertising just by implementing the concept of client profile. In addition to the interactive selection of versioned video segments, the architecture is also designed to allow the insertion of promotional or any other kind of content in the middle of the main streaming playback. Later, the playback can be resumed directly without any kind of offset, interruption or efficiency cost. Hence, our interactive architecture can be extended to offer support to multiple streaming applications. In this paper we focus on adapting broadcasting TV soccer and video surveillance content for smart phone terminals and wireless networks.

Appendix

A. Definition of up-switch thresholds

Let T_0 be the outage probability that the available bandwidth B is greater or equal to twice the bit rate of the video sequence R. We would then look for the deviation threshold d_0 such that

$$P\left[B \geq 2\,R | dev \leq d_0\right] \geq T_0 \tag{5}$$

Conversely, one could set the deviation threshold d_0 and compute the outage probability T_0.

In the `qdisc` set-up, we have measured the deviation in $i = 6$ different scenarii ($B_i \in \{1.2, 1.35, 1.5, 1.7, 1.85, 2\}\,R$). We therefore know

$$P\left[dev \leq d_0 | B = B_i\right] = \int_{-\infty}^{d_0} T_{dev|B=B_i}\left(dev\right) ddev \tag{6}$$

$$= cdf_{dev|B=B_i}\left(d_0\right) \tag{7}$$

We can regard those six scenarii as a sampling of the frequency domain, so as to write the average cdf as

$$\begin{aligned} & P\left[dev \leq d_0\right] \\ = \; & P\left[dev \leq d_0 | B = B_1\right] P\left[B < B_{1,2}\right] \\ & + \sum_{i=2}^{5} P\left[dev \leq d_0 | B = B_i\right] \\ & \quad P\left[B_{(i-1),i} \leq B < B_{i,(i+1)}\right] \\ & + P\left[dev \leq d_0 | B = B_6\right] P\left[B \geq B_{5,6}\right] \end{aligned} \quad (8)$$

where

$$B_{i,j} = \frac{B_i + B_j}{2} \quad (9)$$

Returning to (5), we can write

$$P\left[B \geq 2\,R | dev \leq d_0\right] = 1 - P\left[B < 2\,R | dev \leq d_0\right] \quad (10)$$

This last conditional probability can be transformed thanks to the Bayes formula into

$$\begin{aligned} & P\left[B < 2\,R | dev \leq d_0\right] \\ = \; & \frac{P\left[B < 2\,R\right]\; P\left[dev \leq d_0 | B < 2\,R\right]}{P\left[dev \leq d_0\right]} \end{aligned} \quad (11)$$

In (11), $P\left[B < 2\,R\right]$ depends on the wireless set-up under consideration. Extrapolating from downstream UDP throughput from [47], one could possibly model the available bandwidth from UDP streaming as an exponential distribution parametrised to C, the nominal capacity of the wireless set-up, such that

$$P\left[B < 2\,R\right] = 1 - \exp\left[-\left(\frac{3}{C}\right) 2\,R\right] \quad (12)$$

Based on relations (6-8) and on the bandwidth model (12), we would get

$$\begin{aligned} & P\left[dev \leq d_0 | B < 2\,R\right] \\ = \; & P\left[dev \leq d_0 | B = B_1\right] P\left[B < B_{1,2}\right] \\ & + \sum_{i=2}^{5} P\left[dev \leq d_0 | B = B_i\right] \\ & \quad P\left[B_{(i-1),i} \leq B < B_{i,(i-1)}\right] \\ & + P\left[dev \leq d_0 | B = B_6\right] P\left[B_{5,6} \leq B < 2\,R\right] \end{aligned} \quad (13)$$

$$\begin{aligned} = \; & \left\{1 - \exp\left[-\left(\tfrac{3}{C}\right) 1.275\,R\right]\right\} cdf_{dev|B=B_1}\left(d_0\right) \\ & + \left\{ \begin{array}{l} \exp\left[-\left(\tfrac{3}{C}\right) 1.275\,R\right] \\ -\exp\left[-\left(\tfrac{3}{C}\right) 1.425\,R\right] \end{array} \right\} cdf_{dev|B=B_2}\left(d_0\right) \\ & + \left\{ \begin{array}{l} \exp\left[-\left(\tfrac{3}{C}\right) 1.425\,R\right] \\ -\exp\left[-\left(\tfrac{3}{C}\right) 1.6\,R\right] \end{array} \right\} cdf_{dev|B=B_3}\left(d_0\right) \\ & + \left\{ \begin{array}{l} \exp\left[-\left(\tfrac{3}{C}\right) 1.6\,R\right] \\ -\exp\left[-\left(\tfrac{3}{C}\right) 1.775\,R\right] \end{array} \right\} cdf_{dev|B=B_4}\left(d_0\right) \\ & + \left\{ \begin{array}{l} \exp\left[-\left(\tfrac{3}{C}\right) 1.775\,R\right] \\ -\exp\left[-\left(\tfrac{3}{C}\right) 1.925\,R\right] \end{array} \right\} cdf_{dev|B=B_5}\left(d_0\right) \\ & + \left\{ \begin{array}{l} \exp\left[-\left(\tfrac{3}{C}\right) 1.925\,R\right] \\ -\exp\left[-\left(\tfrac{3}{C}\right) 2\,R\right] \end{array} \right\} cdf_{dev|B=B_6}\left(d_0\right) \end{aligned} \quad (14)$$

$$\begin{aligned} & P\left[dev \leq d_0\right] \\ = \; & P\left[dev \leq d_0 | B < 2\,R\right] \\ & + cdf_{dev|B=B_6}\left(d_0\right) P\left[B \geq 2\,R\right] \end{aligned} \quad (15)$$

$$\begin{aligned} = \; & P\left[dev \leq d_0 | B < 2\,R\right] \\ & + \exp\left[-\left(\frac{3}{C}\right) 2\,R\right] cdf_{dev|B=B_6} \end{aligned} \quad (16)$$

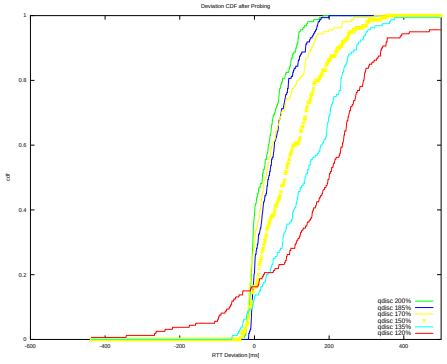

Figure 18. CDFs for the six scenarios

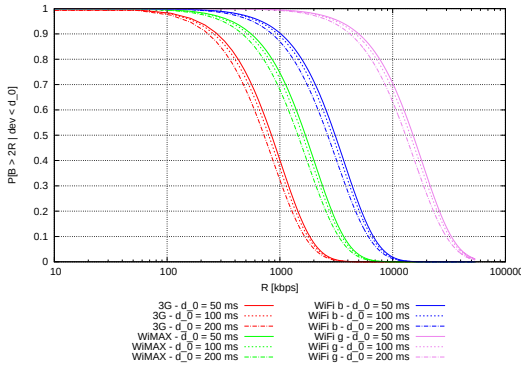

Figure 19. Probability that there is enough bandwidth to upswitch s.t. observed deviation. C is worth respectively 3 Mbps (3G), 6 Mbps (IEEE 802.16 - WiMAX), 11 Mbps (IEEE 802.11b - WiFi) and 54 Mbps (IEEE 802.11g - WiFi)

Looking at Fig. 18, we can measure Table VII:

Margin	50 ms	100ms	200ms
1.2	.2	.3	.5
1.35	.3	.4	.7
1.5	.4	.6	.9
1.7	.6	.75	.95
1.85	.6	.8	1
2	.7	.85	1

TABLE VII.
DEVIATION SAMPLES FROM FIG. 18

Injecting values of Table VII into relations (14) and (16), we can plot the probability (10) in Fig. 19. For a deviation $d_0 = 50$ ms, an upswitch has a 90% success rate provided the streaming rate R is lower than 300 kbps in 3G networks, 500 kbps in WiMAX scenario, 1 Mbps in WiFi b and 5 Mbps in WiFi g. Considering a sequence at 1 Mbps streamed on a 3G network, the upswitch has a 30% success rate if the observed deviation is up to 200 ms, whereas this rate increases to 40% if the deviation is as low as 50 ms.

REFERENCES

[1] A. Argyriou and V. Madisetti, "Streaming h.264/avc video over the internet," in *Consumer Communications and Networking Conference*, Jan. 2004, pp. 169–174.

[2] M. F. Sayit and T. Tunah, "Video streaming with h.264 over the internet," in *Signal Processing and Communications Applications*, Apr. 2006, pp. 1–4.

[3] Z. Li and Z. Zhang, "Real-time streaming and robust streaming h.264/avc video," in *Third International Conference on Image and Graphics*, Dec. 2004, pp. 353–356.

[4] J.Lu, G.Lafruit, and F.Catthoor, "Fast reliable multi-scale motion region detection in video processing," in *ICASSP*, vol. 1, April 2007, pp. 689–692.

[5] P.Baccichet, X.Zhu, and B.Girod, "Network-aware h.264/avc region-of-interest coding for a multi-camera wireless surveillance network," in *Picture Coding Symp.*, April 2006.

[6] T. Bae, T. C.Thang, D.Y.Kim, Y.M.Ro, J.W.Kang, and J.G.Kim, "Multiple region-of-interest support in scalable video coding," in *ETRI Journal*, vol. 28, April 2006, pp. 239–242.

[7] C. DeVleeschouwer, T. Nilsson, K. Denolf, and J. Bormans, "Algorithmic and architectural co-design of a motion-estimation engine for low power video devices," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 12, Dec. 2002.

[8] R. Sutter, K.DeWolf, S.Lerouge, and R.VandeWalle, "Lightweight object tracking in compressed video streams demonstrated in region-of-interest coding," *EURASIP*, pp. 59–75, January 2007.

[9] A. Senior, A. Hampapur, Y.-L. Tian, L. Brown, S. Pankanti, and R. Bolle, "Appearance models for occlusion handling," *Image and Vision Computing*, vol. 24, no. 11, pp. 1233–1243, November 2006.

[10] J. You, G. Liu, and L. Sun, "A multiple visual models based perceptive framework for multilevel video summarization," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 3, pp. 273–285, March 2007.

[11] A.Cavallaro, O.Steiger, and T.Ebrahimi, "Semantic video analysis for adaptive content delivery and automatic description," in *IEEE Tran. on CSVT*, vol. 15, October 2005, pp. 1200–1209.

[12] A. G. Money and H. Agius, "Video summarization: a conceptual framework and survey of the state of the art," *Journal of Visual Communication and Image Representation*, vol. 19, pp. 121–143, 2008.

[13] B. T. Truong and S. Venkatesh, "Video abstraction: A systematic review and classification," in *ACM Transactions on Multmedia Computing, Communication and Applicationl*, vol. 3, 2007.

[14] A. Mavlankar and B. Girod, "Spatial-random-access-enabled video coding for interactive virtual pan/tilt/zoom functionality," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 21, no. 5, pp. 577–588, 2011.

[15] I. A. Fernandez, F.Chen, F.Lavigne, X.Desurmont, and C. DeVleeschouwer, "Browsing sport content through an interactive h.264 streaming session," in *MMEDIA*, Athens, June 2010.

[16] I. A. Fernandez, F.Lavigne, X.Desurmont, and C. DeVleeschouwer, "Worthy visual content on mobile through interactive video streaming," in *ICME:2010 IEEE International Conference on Multimedia and Expo*, Singapore, July 2010.

[17] W. You, M. S. H. Sabirina, and M. Kima, "Real-time detection and tracking of multiple objects with partial decoding in h.264/avc bitstream domain," in *SPIE*, vol. 7244, Feb. 2009.

[18] W.Wang, J.Yang, and W.Gao, "Modeling background and segmenting moving objects from compressed video," *IEEE transactions on circuits and systems for video technology*, vol. 18, pp. 670–681, May 2008.

[19] A. Gyaourova, C. Kamath, and S.-C. Cheung, "Block matching for object tracking," *University of California Radiation Laboratory-TR*, vol. 200271, October 2003.

[20] D.J.Thirde, M.Borg, V.Valentin, L. Barthelemy, J.Aguilera, G.Fernandez, J.M.Ferryman, F.Bremond, M.Thonnat, and M.Kampel, "People and vehicle tracking for visual surveillance," in *VS 06: Proceedings of the IEEE International Workshop on Visual Surveillance*. ACM, Mai. 2006.

[21] X. Yu and D. Farin, "Current and emerging topics in sports video processing," in *IEEE International Conference on Multimedia and Expo (ICME)*, 2005.

[22] Y. Takahashi, N. Nitta, and N. Babaguchi, "Video summarization for large sports video archives," *Multimedia and Expo, IEEE International Conference on*, vol. 0, pp. 1170–1173, 2005.

[23] L. Sun and G. Liu, "Field lines and players detection and recognition in soccer video," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASPP)*, 2009, pp. 1237–1240.

[24] D. Tjondronegoro, Y. Chen, and B. Pham, "Highlights for more complete sports video summarization," *IEEE Transactions on Multimedia*, vol. 11, pp. 22–37, 2004.

[25] ITU-T, "H.264: Advanced video coding for generic audiovisual services," *Series H : Audiovisual and multimedia systems*, 2005.

[26] H. Schulzrinne, A. Rao, and R. Lanphier, "Real time streaming protocol (rtsp)," RFC 2326 (Proposed Standard), Apr. 1998. [Online]. Available: http://www.ietf.org/rfc/rfc2326.txt

[27] A. Z. Sarker, "A study over adaptive real time video over lte," in *Ph.D. thesis*, Lulea University of Technology, 2007.

[28] T. Schierl and T. Wiegand, "H.264/avc rate daptation for internet streaming," in *14th International Packet Video Workshop (PV)*, Irvine, CA, USA, December 2004.

[29] I. RealNetworks, "Helix mobile server rate control for mobile networks," 2008.

[30] H. Schulzrinn, S. Casner, R. Frederick, , and V. Jacobson, "Rtp: A transport protocol for real-time applications," in *Tech. Rep., IETF RFC 3550*, July 2003.

[31] V. Paxon and M. Allman, "Computing tcps retransmission timer," in *Tech. Rep., IETF RFC 2988*, November 2000.

[32] M. P. Farrera, M. Fleury, K. Guild, and M. Ghanbari, "Measurement and analysis study of congestion detection for internet video streaming," in *Journal of Communications, vol. 5, no. 2, pp. 169–177*, February 2010.

[33] J. Fabini, W. Karner, L. Wallentin, and T. Baumgartner, "The Illusion of Being Deterministic -Application-Level Considerations on Delay in 3G HSPA Networks," in *NETWORKING '09: 8th International IFIP-TC 6 Networking Conference*. Berlin, Heidelberg: Springer-Verlag., 2009, pp. 301–312.

[34] P. R. Maierhofer, F. Ricciato, A. D'Alconzo, R. Franzan, and W. Karner, "Network-wide measurements of tcp rtt in 3g," in *TMA 09: First International Workshop on Traffic Monitoring and Analysis*. Berlin, Heidelberg: Springer-Verlag., 2009, pp. 17–25.

[35] K. Jang, M. Han, S. Cho, H.-K. Ryu, J. Lee, Y. Lee, and S. Moon, "3g and 3.5g wireless network performance measured from moving cars and high-speed trains," in *MICNET '09 Proceedings of the 1st ACM workshop on Mobile internet through cellular networks*, Beijing, China, 2009.

[36] W. Eklof, "Adapting video quality to radio links with different characteristics," in *Master of Science Thesis*, Sweden, 2008.

[37] J. Navratil and R. L.Cotrell, "Abwe: A practical approach to available bandwidth estimation," in *Stanford Linear Accelerator Center (SLAC)*, 2003.
[38] V. Ribeiro, R. Riedi, R. Baraniuk, J. Navratil, and L. Cotrell, "pathchirp: Efficient available bandwidth estimation for network paths," in *Passive and Active Measurement Workshop*, 2003.
[39] M. Li, M. Claypool, and R. Kinicki, "Wbest: a bandwidth estimation tool for ieee 802.11 wireless networks," in *In Proceedings of 33rd IEEE Conference on Local Computer Networks (LCN)*, Montreal, Quebec, Canada, October 2008.
[40] I. A. Fernandez, P.R.Alface, T.Gan, R.Lauwereins, and C. DeVleeschouwer, "Integrated h.264 region-of-interest detection, tracking and compression for surveillance scenes," in *PV 10: 18th International Packet Video Workshop*, Hong Kong, December 2010.
[41] X. Desurmont, A. Bastide, J. Czyz, C. Parisot, J.-F. Delaigle, and B. Macq, "A general purpose system for distributed surveillance and communication," in *Intelligent Distributed Video Surveillance Systems*. S.A Velastin and P Remagnino Eds, 2006.
[42] F. Chen and C. D. Vleeschouwer, "A resource allocation framework for summarizing team sport videos," in *IEEE International Conference on Image processing(ICIP), Cairo, Egipt*, 2009.
[43] J. Li, T. Wang, W. Hu, M. Sun, and Y. Zhang, "Soccer highlight detection using two-dependence bayesian network," in *IEEE International Conference on Multimedia and Expo (ICME)*, 2006.
[44] T.Schierl, T. Wiegand, and M. Kampmann, "3GPP Compliant Adaptive Wireless Video Streaming Using H.264/AVC," in *IEEE International Conference on Image Processing. ICIP 2005.* .
[45] R. Gass and C. Diot, "An experimental performance comparison of 3g and wi-fi," in *11th Passive and Active Measurement Conference (PAM 2010)*, Zurich, 2010.
[46] G. Gommand, L.Schumacher, and G. Toma, "Performance Evaluation of Indoor Internet Access over a Test LTE Mini-Network," in *The 14th International Symposium on Wireless Personal Multimedia Communications, WPMC'11* , October 2011.
[47] A. Balasubramanian, R. Mahajan, and A. Venkataramani, "Augmenting mobile 3g using wifi: Measurement, system design and implementation," in *MobiSys 2010*, San Francisco, USA, 2010.

Ivan Alen Fernandez received his M.Sc. Telecommunications Engineering degree from the University of Vigo (Spain), in 2009.

As a part of his studies, he developed his Master Thesis on video coding (H.264) and motion detection & tracking in the multimedia group of the research institute IMEC (Belgium) in 2008-2009. He was research assistant at the Université Catholique de Louvain (UCL), between 2009 and 2011. His main interest topics include video networking, security and cryptography. He has also worked at Vodafone Spain as trainee in 2008 in the Radio Networks Department and currently he is Quality Control Manager in critical software development (Daintel) for ICUs in Denmark.

Christophe De Vleeschouwer received the Electrical Engineering degree and the Ph. D. degree from the Université Catholique de Louvain (UCL) Louvain-la-Neuve, Belgium in 1995 and 1999 respectively.

He is currently a permanent Research Associate of the Belgian NSF and an assistant professor at UCL. He was a senior research engineer with the IMEC Multimedia Information Compression Systems group (1999-2000), and contributed to project with ERICSSON. He was also a post-doctoral Research Fellow at UC Berkeley (2001-2002), and at EPFL (2004). His main interests concern video and image processing for communication and networking applications, including content management and security issues. He is also enthusiastic about non-linear signal expansion techniques, and their use for signal analysis and signal interpretation. He is the co-author of more than 20 journal papers or book chapters. He did coordinate the FP7-216023 APIDIS European project (www.apidis.org), and several Walloon region projects, respectively dedicated to video analysis for autonomous content production, and to personalized and interactive mobile video streaming.

George Toma received his engineer degree at the Polytechnic University of Bucharest, Romania in automatic control, in 2007.

He is currently a PhD student at FUNDP - The University of Namur, Belgium. He was a research assistant at the same university between 2007 and 2010. His research topics include adaptive streaming techniques, quality assessment of streaming sessions and performance evaluation of wireless access networks.

Laurent Schumacher received his M.Sc. EE. Degree from the Faculté Polytechnique de Mons, Belgium, in 1993 and his Ph.D. degree from the Université catholique de Louvain, Belgium, in 1999. Since 2003, he has been a professor at FUNDP - The University of Namur, Belgium, after a post-doctoral stay at Aalborg Universitet, Denmark. His current research interests include performance evaluation of cellular systems (LTE and beyond) and SIP/IMS signalling. Prof. Schumacher is a member of the IEEE Computer Society and the ACM.