A Feedback-Based Control Technique for Interactive Live Streaming Systems to Mobile Devices

Gianluca Paravati, Cesare Celozzi, Andrea Sanna, Fabrizio Lamberti, *Member, IEEE*

Abstract — *Recent advances in wireless communication have raised possibility of creating new sets of richer multimedia applications in wireless networks; on the other hand, despite the rapid evolution and growing popularity of mobile handheld devices such as PDAs and smart phones, they are still significantly computationally constrained by limited resources for some kind of applications (e.g. computer graphics).*

The main contribution of this work is the design of a closed-loop controller in the context of image/video streaming systems with interactivity constraints to automatically adapt streaming parameters to bandwidth fluctuations. The proposed technique is general and acts as a sort of Quality of Service (QoS) mechanism at the application level and can be applied to any kind of client, but in this context we point out at common handheld devices, such as PDAs and smart phones, as a target. Experiments were carried out using different encoding standards (M-JPEG and MPEG-4 Part 2) and different network access (802.11g and HSDPA)[1].

Index Terms — Remote Rendering, Interactive Live Streaming, Controller, Mobile Devices.

I. INTRODUCTION

Most of existing live streaming systems for multimedia applications do not have strict requirements in terms of latency because the interaction feature is limited to playback. Examples are video conference and video on demand (VOD). On the other hand, streaming systems for interactive use, i.e. structured to receive as far as possible graphical content as a result of user interactions, need a very low delay between the user feedback command and its effect (round-trip time - RTT), otherwise they shall not be usable [1]. Examples are, for instance, remote control [2][3][4], tele-surgery[5], and streaming game technologies[6][7]. Indeed, recently streaming-based technologies applied to the gaming scenario are becoming very attractive; a number of commercial solutions offering the possibility of gaming without the need for specialized graphics hardware at anytime, everywhere, are becoming to appear on the gaming market (on-demand game platform such as OnLive, Intel StreamMyGame, Vollee, www.onlive.com). Moreover, these new development trends for game developers are of great interest because they reflect upon new business distribution models, such as the Play to Play and Software as a Service (SaaS) [8]. These platforms are based on remote rendering approaches [9][10][11][12][13], where a rendering server is in charge to create, compress and stream to a client the graphical representation of the processed data, thus moving the computational complexity from the client to the server side (this ubiquitous paradigm allows users to use computationally expensive video games also on mobile devices).

Real-time streaming technologies have to deal with a lot of issues [14][15], mainly related to latency, round trip time (time elapsed between sending a command and observing its effects), encoding complexity, bandwidth capacity and bandwidth fluctuations. In particular, bandwidth fluctuations can be caused by network links congestion and, especially for mobile devices, by low signal in fringe coverage areas. Many conventional systems, e.g. H.264 [16], use buffering in order to avoid playback break but this technique intrinsically increases the delay; on the other hand, interactive live streaming applications care mostly about updated contents. A flow of compressed still images, e.g. M-JPEG [17], has proven to be effectively used in order to keep very low latencies and low processor overhead at the expense of increased bandwidth [18]. M-JPEG encoding parameters that can be tuned are resolution (R), frame rate (F) and image quality (Q) [19]. Depending on the encoding parameters, a mobile device can be able or not to fully decode the incoming data flow. For example, the higher is the resolution, the higher is the computational time needed to decode a single frame; similar considerations can be made for image quality and generated frame rate.

Bandwidth fluctuations have an impact on the amount of information that can reach a destination in the unit of time; as a consequence, a trade-off among the above streaming parameters has to be reached, because bandwidth capacity may not be sufficient to guarantee a smooth and detailed visualization, thus affecting the overall user satisfaction (i.e. the perceived performance).

Internet streaming applications usually encode the same video at different bit-rate levels and the decoder tries to gain access to the flow with the highest bit-rate. A higher bit-rate provides a higher visual quality, but if the communication channel conditions do not allow the use of the highest bit-rate, applications commute to the immediately lower bit-rate flow.

This paper proposes a system aimed to automatically optimize the selection of video encoding parameters basing on a closed-loop controller in such a way mobile users can easily and smoothly visualize interactive multimedia content. Parameters are continuously tuned during the streaming in

[1] G. Paravati, C. Celozzi, A. Sanna, and F. Lamberti are with the Dipartimento di Automatica e Informatica, Politecnico di Torino, I-10129 Torino, Italy (e-mail: gianluca.paravati@polito.it, cesare.celozzi@polito.it, andrea.sanna@polito.it, fabrizio.lamberti@polito.it).

Manuscript received January 15, 2010

order to exploit both the decoding capabilities of the mobile device and the current status of the underlying network infrastructure. The system has been implemented and tested using the M-JPEG encoding. Since the challenge is to have smooth and interactive remote rendering for handheld devices, we care about frame rate and round-trip time. The same principle can be used elsewhere in another video codec; to serve this scope, several tests have been performed using the MPEG-4 video encoding. Preliminary experiments obtained setting up a live video streaming application for mobile devices shows the feasibility of the proposed approach.

The main features of the proposed control based approach are:

- Customization to end device capabilities.
- Mitigation of bandwidth fluctuation effects, thus providing an optimal trade-off among live encoding parameters.
- Feedback of the device throughput that allows the controller to dynamically compensate for disturbances basically due to bandwidth fluctuations and device capabilities.

The organization of this paper is given as follows. Section II provides a detailed description of the control system. Experimental tests and results are illustrated in Section III. Finally, conclusion and future works can be found in Section IV.

II. THE CONTROL SYSTEM

The proposed system aims to take advantage of automatic control techniques able to concurrently and smoothly tune all the parameters involved in M-JPEG video streaming without a-priori knowledge about the precise effects caused by the modification of these parameters. Control techniques have been already successfully applied in the past to ensure QoS of networked systems [20]. In a M-JPEG streaming scenario it is possible to identify a set of parameters influencing the visualization performance:

- resolution;
- frame rate;
- image quality.

The relationship among these parameters [19] and the expected theoretical bandwidth occupation (TBO) can be modeled as:

$$TBO(t) = \frac{f(t) \cdot w(t) \cdot h(t) \cdot C_d}{C_r(t)} \quad (1)$$

where f(t) is the frame rate, w(t) and h(t) are the resolution parameters (width and height), C_d is the color depth expressed in bit per pixels and $C_r(t)$ is the compression ratio. All these parameters are time dependent because they can change from time to time, but the color depth C_d that is, in general, fixed to 16 or 24 bpp. In this way, depending on: the current frame rate generated by the server, the resolution of the encoded stream and the M-JPEG quality factor (strictly related to compression ratio) it is possible to know in advance the bandwidth occupation of the streaming flow.

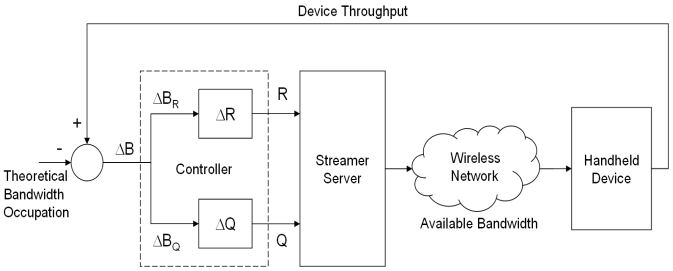

Fig. 1. The closed-loop controller system

The designed controller deals with the bandwidth control trying to stabilize the system around a target frame rate that is specified by the user. The theoretical bandwidth occupation is directly proportional to the visualization frame rate.

The proposed system is depicted in Figure 1. It is composed by a controller (the dashed box), a streamer server and a common handheld device. The controller takes as input the difference between a performance measure, taken at the handheld device, and the reference of the system, later described in detail. The controller computes and provides the streamer server the encoding parameters, that is the resolution R and the image quality Q.

The reference of the controller is the theoretical bandwidth occupation TBO computed as (1). The measured output of the whole system is the Device Throughput (DT); it is periodically measured by the handheld device and it represents the amount of visualization flow that can be decoded by the device in the unit of time. If the period of this measure is reduced, it is possible to improve the responsiveness of the system to changes in network conditions (i.e. how quickly the system responds to bandwidth variations). The client periodically feeds this data back to the controller module, as a sort of receiver report. In this way, the input of the controller becomes the bandwidth error ΔB, computed as:

$$\Delta B(t) = DT(t) - TBO(t) \quad (2)$$

The bandwidth error ΔB represents the difference between the visualization flow generated by the live streaming server and the actual visualization flow that the handheld device is able to receive, decode and display in the unit of time. If the device throughput is lower than the theoretical bandwidth occupation, it means that the handheld device is not able to process all the visualization flow sent by the server. The feedback coming back from the device allows the controller to dynamically compensate for disturbances to the system due to bandwidth fluctuations and insufficient device capabilities.

The controller is in charge to continuously adjust the control parameters, in this case resolution and image quality, as

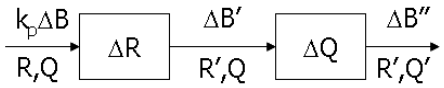

Fig. 2. SISO model of the controller

necessary to keep the input error ΔB to a minimum, i.e. minimizing the bandwidth error, thus the live streaming is tailored to best suit both current network conditions and the device throughput. The basic idea is to split up the controller error among the considered parameters; different parts of the bandwidth error are used to tune each encoding parameter. In other words, the adapted parameters undergo changes that are proportional to the input error, then they are used by the live streaming server to create, compress, and stream the image representation of a 3D scene to the mobile device.

The considered parameters present reciprocal interactions, meaning that the concurrent modification of each of them has an indirect influence on the input error of the controller; e.g. a slight change of the encoding parameter 'resolution' (image width and height) modifies the theoretical bandwidth occupation TBO (it introduces a quadratic contribution) and the bandwidth error ΔB changes as a matter of fact. Thus, parameters cannot change at the same time and this fact suggests to split the control phase in cascade stages, as shown in Fig. 2 where the Single Input Single Output model of the designed controller is illustrated. Each stage is in charge to tune a different variable. The input of each stage is always a bandwidth quantity. Each stage exploits the bandwidth quantity provided by the previous stage in order to allow the modification of the variable to be controlled; the output bandwidth of each stage onwards reduces till exhausting.

In the SISO model of Fig. 2, the first stage receives as input the bandwidth error and uses part of it to regulate the resolution parameter according to the following equation:

$$\Delta R = \frac{k_p \Delta B}{B_R^{up} - B_R} \tag{3}$$

where ΔR is the amount of increment or decrement computed for the resolution index; the resolution can assume discrete values in a given ordered set, e.g. from 240x180 to 400x300 pixels, and for each couple of values *width* × *height* corresponds a resolution index, thus creating different levels of resolutions. k_p is a proportionality factor and ΔB is the bandwidth error with respect to the theoretical bandwidth occupation. B_R^{up} is the theoretical bandwidth occupation computed with the current resolution index of the streaming. Finally, B_R is the theoretical bandwidth occupation computed with the immediately upper resolution index with respect to the current resolution index. When the resolution parameter reaches the maximum available level, it is not possible to compute the bandwidth required to increase it again, thus the following equation is used:

$$\Delta R = \frac{k_p \Delta B}{B_R - B_R^{low}} \tag{4}$$

where B_R^{low} is the theoretical bandwidth occupation computed with the immediately lower resolution index of the streaming.

The new resolution value is adopted as a new encoding parameter after the rendering of the next frame. The modification of this parameter causes a change in theoretical bandwidth occupation. Thus, the theoretical bandwidth occupation is recomputed, yielding to a new bandwidth error ($\Delta B'$) that will be taken as input of the second stage of the controller that is in charge to tune the second parameter according to the following equation:

$$\Delta Q = k_q \cdot \left(1 - k_p\right) \Delta B' \tag{4}$$

where ΔQ is the amount of increment or decrement concerning the image quality index and k_q is a proportionality coefficient.

The algorithm that performs the closed-loop control can be summarized as follows:

1. The control system receives the feedback measure (device throughput) from the client and evaluates the gap ΔB with respect to the theoretical bandwidth occupation.
2. Part of the input error is "assigned" to the resolution parameter, which can be correspondingly tuned by means of an appropriate bandwidth allocation.
3. The quality index is smoothly tuned on the basis of the bandwidth error sign.
4. The controller conveys decisions about the parameters to be provided the streamer server, that consequently bundles the live stream.

The handheld device receives the stream and measures the device throughput which is then delivered back to the controller. This measure allows the system to take into account of bottlenecks due to limited available channel bandwidth and computational limitations of the (thin) client device. A similar approach can be used in a MPEG streaming scenario.

III. TESTS AND RESULTS

The scope of the current work is to design a controller that automatically tunes encoding parameters in a live video streaming system to mobile devices in order to adapt the visualization flow independently of the network status. The system has been implemented and tested with different network connections. The streaming system is composed by a rendering and streaming server and a mobile device. The rendering server has been written in C++ language and it is in

Fig. 3. The remote rendering system

charge to render 3D scenes on behalf of the mobile device. After the rendering of a frame, the frame buffer of the server is compressed and streamed to the client device. The server is composed by Dual-Core AMD Opteron CPU 2.60 GHz, 4 GB of RAM and a NVIDIA Quadro FX 3500 graphics card. The client application has been written from scratch and runs on every mobile phone or device with a Java Micro Edition virtual machine and access to Internet network connection; most of the smart phones and PDAs, or almost all, currently available on the market are in posses of these features. The client application is in charge to establish a connection with the rendering server, receive, decode and display the stream representing the 3D scene visualization. Moreover commands can be sent from the mobile device to the rendering server to change the point of view of the virtual camera and navigate in the 3D virtual world. It is worthwhile to notice that the proposed control system can be applied to any kind of live streaming scenario, not only 3D remote visualization (3D visualization has been chosen as it needs high interaction levels). Since the real test scenario of this work is 3D visualization, i.e. a highly interactive application, a crucial parameter is the round trip time computed as the elapsed time between sending a command (e.g. a roto-translation) from the mobile device to the rendering server and showing on the screen of the mobile terminal the first frame corresponding to the effect of the user command. Results of the tests have been gathered using a real commercially available device.

This section first analyzes the controller behavior under varying simulated network conditions to point out how M-JPEG encoding parameters change during a visualization session on a real mobile device connected to the server through a WiFi (802.11g) access point.

The same remote rendering scenario has been implemented and tested also using a MPEG video coding scheme. From the mobile device point of view, the difference with respect to the first scenario is due to the fact that it is possible to use built-in video streaming applications, available on most smart phones and PDAs. Currently, only the MPEG live streaming part of

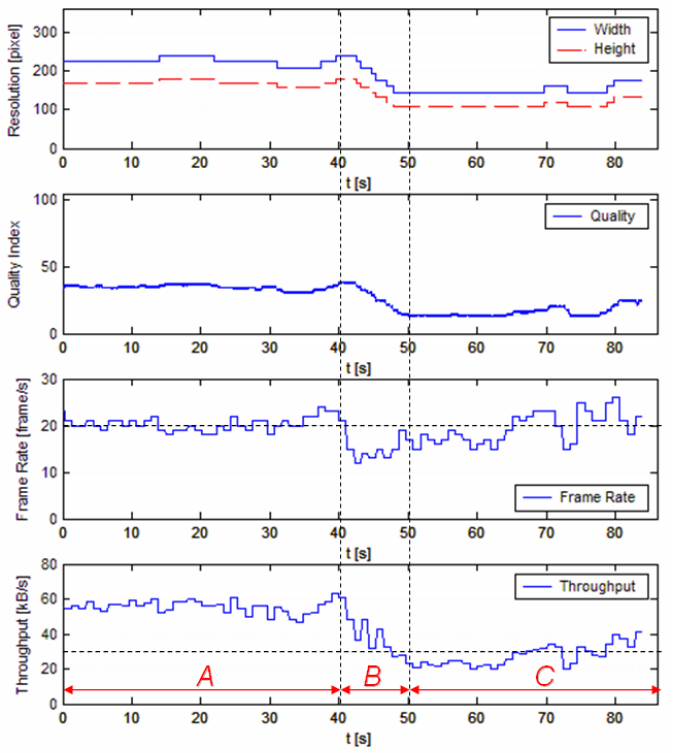

Fig. 4. Controller behavior during a M-JPEG live streaming to a mobile device with varying network conditions.

the system has been tested; conversely to the M-JPEG scenario, the full automatic control functionality is yet to be implemented. However, in the MPEG scenario we have the possibility of changing on-the-fly a subset of the encoding parameters to conduct performance evaluations. By using built-in commercial players for the visualization phase, it would be possible to implement the proposed controller by developing a probe enabling measures that are analogous to the ones used in the M-JPEG scenario. Preliminary results related to the MPEG live video streaming are shown in the following of this Section. Tests are aimed to investigate the round-trip time and the influence of MPEG encoding parameters on the bandwidth usage.

Figure 3 shows the implemented remote rendering system during a visualization session. The server, shown in the background, is in charge to stream the live content to the mobile device (foreground).

A. Controller Behavior

The described controller scheme has been evaluated from a behavioural point of view to show the effectiveness of the proposed approach.

Figure 4 shows results obtained by transmitting a M-JPEG stream to a real handheld device. The mobile client is connected through a 802.11g wireless access point. From top to bottom the monitored parameters are: resolution, image quality, actual frame rate reached by the mobile device, and the measured throughput (i.e. the feedback feature).

The graphical user interface of the mobile device allows the user to set a target frame rate, i.e. the frame rate desired by the user. If the current configuration of the encoding parameters does not allow the frame rate at the client side to reach this value, the system will try to modify them in order to follow the target value. It is worth observing that the frame rate directly influences the bandwidth occupation as in (1).

At the beginning of the test, the system has to maintain a constant frame rate set to 20 fps. During the first part of the simulation (period A, between 0 and 40 s) the system worked in stationary conditions, where no bandwidth limitation was imposed; as a result, the system was steady around quality and resolution values that allowed to exploit the maximum throughput of the device (around 60 KB/s).

At the beginning of period B (at the time t = 40 s), the available output bandwidth was consciously limited to 30 KB/s to simulate a bottleneck. The bandwidth limitation is simulated using a network limiter installed on the rendering server machine. As a result, the frame rate rapidly dropped down because the bandwidth was not enough to allow the transmission of the image flow with the previously used parameters. As a consequence, the controller system reacted gradually reducing the parameter values; it is possible to observe that both the resolution and the image quality index are automatically reduced.

During the period C, the controller leaded the system to a different steady state, rising the frame rate, thus mitigating the effects of the bandwidth bottleneck previously introduced. The frame rate increases because the parts of the bandwidth devoted to the resolution and quality parameters decrease, allowing the streamer to send a greater number of frames.

B. MPEG-4 Live Streaming

The proposed control methodology is more general and could be applied to video compression schemes such as MPEG, thus providing an effective solution for consumer devices that natively support this kind of encodings. In this Section, a preliminary study of the system under analysis using the MPEG-4 Part 2 video codec is presented.

MPEG-4 has been chosen among the main video codecs because it deals with video encodings supported by the majority of consumer devices. Moreover, it lends itself well to live streaming scenarios since it is not expensive from a computational point of view for both the server, that is in charge to encode, and the (thin) client device that reproduces the video stream.

Results of the tests illustrated in this Section are gathered using a reflected stream method. Starting from the raw frames computed by the rendering server, an MPEG4 encoder generates the video flow that is subsequently made available to mobile users through a streaming server. An RTP session is established for each live streaming request; it is used in conjunction with the RTCP protocol to monitor transmission statistics. The video streaming transmission is based on the UDP protocol and the ports which "form" a session are negotiated using the RTSP protocol (using the SDP protocol during the setup phase). Encoding functionalities are provided by the open source VideoLAN (VLC) media player (http://www.videolan.org/), whilst streaming functionalities are provided by the open source Darwin Streaming Server (http://developer.apple.com/opensource/server/streaming/index.html). VLC media player uses the open source library of codecs FFmpeg (http://ffmpeg.org/) for the encoding phase. A run-time modification of the MPEG encoding parameters used by this library makes possible both creating the live video streaming and modifying the output bit rate. The output format of the video stream is variable bit-rate.

In this scenario, a set of parameters has been tested to change:

- the minimum macro block quantizer (q_{min});
- the maximum macro block quantizer (q_{max});
- the GOP size (group of pictures).

Preliminary results are evaluated in terms of bandwidth occupation and round-trip time.

The quantization offset is defined as the difference between the two extreme values (q_{max}-q_{min}). The minimum quantization offset and the smallest values of q_{max} and q_{min} (respectively 1 and 2) correspond to best visualization quality. The result of the modification of these encoding parameters is a lower visualization quality and a lower bit rate produced by the encoder. In order to study the effects that the modification of this parameter has on the bandwidth, two cases are investigated:

- quantizer values variation with fixed quantization offset;
- quantizer values variation with variable quantization offset.

Figure 5 shows the influence of quantizer values on bandwidth requirements. Plotted lines represent the average bit rate trend with different quantizer values, that are modified by keeping fixed the quantization offset. Each line represents a live streaming session with fixed quantizer values, i.e. they are not modified on-the-fly. During a live streaming session, a 3D scene is shown on the mobile device (like the one shown in Figure 3). It is clearly visible that the required bandwidth halves with respect to the best visualization quality line (corresponding to $q_{min}=1$, $q_{max}=2$) when quantizer values becomes $q_{min}=4$, $q_{max}=5$. The required bandwidth is less than 1/3 with $q_{min}=7$, $q_{max}=8$. Simulations with quantizer values greater than 30 are not of interest because the visualization is impaired by artifacts.

The encoder module has been modified in order to allow on-the-fly changes of quantizer values. Figure 6 shows results gathered during a live streaming session where quantizer values are allowed to change during the test; again, quantization offset is kept fixed. The simulation has been divided into regular intervals, that are identified by two values, q_{min} and q_{max} respectively; e.g. during the first period of time quantizer values were $q_{min}=1$, $q_{max}=2$. After every regular interval quantizer values are increased; as a result, a decrease in bandwidth occupation is pointed out. The trend observed is a descending exponential. Since the run-time modification of the above encoding parameters causes a change in bandwidth requirements, this graph demonstrate the feasibility of the proposed approach also to the MPEG live streaming scenario.

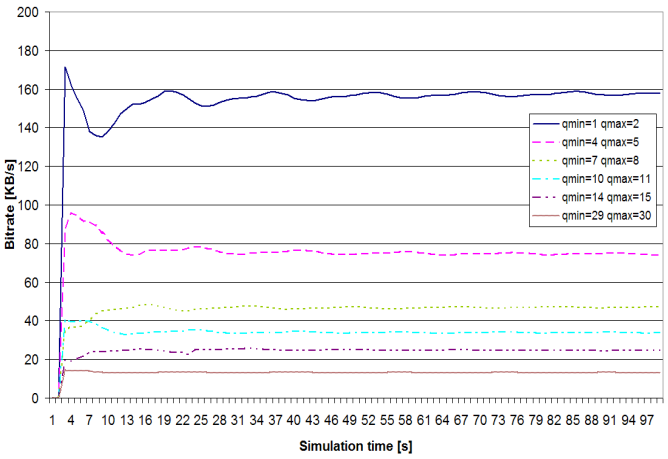

Fig. 5. Bandwidth measurements by varying quantizer values (qmin, qmax) in different live streaming sessions. The quantization offset is kept fixed.

Figure 7 shows bandwidth trends as the quantization offset changes; moreover, it shows how changes in minimum quantizer values impacts on bandwidth occupation. Each curve represents the bandwidth occupation during a simulation test where the minimum macro block quantizer value q_{min} is fixed and the maximum macro block quantizer value q_{max} increases at regular intervals. Each interval is identified by the q_{max} value. As the quantization offset increases, the measured bandwidth decreases. Similar considerations can be made observing curves with increasing minimum quantizer values: again, the measured bandwidth decreases; e.g. the bandwidth occupation halves passing shifting from $q_{min}=1$ to $q_{min}=5$. Also this graph demonstrates the feasibility of the proposed approach to the MPEG live streaming scenario.

Moreover, other experiments have been performed to investigate the relationship between the GOP size and the bandwidth occupation; by varying, at the run-time, the GOP size, it has not been noticed a significant change in bandwidth requirements.

C. M-JPEG vs. MPEG

The control algorithm can be adopted both in M-JPEG and MPEG live video streaming techniques, thus covering a wide range of consumer devices. However, these encodings introduce different time requirements due to the buffering process. In this study, interactivity has been studied in terms of round-trip delay time, i.e. the elapsed time between sending a command from the mobile device to the rendering server and receiving back the first frame where the command has been applied.

Table I and Table II indicate the average round-trip time experienced respectively with M-JPEG and MPEG-4 encoding during the tests. Moreover, results have been gathered for two different types of available connection protocols: WiFi 802.11g and HSDPA (3.6Mbps). In the first case, the mobile device is connected through a wireless access point to the same LAN of the rendering server, whereas, in the second case, it is connected to a 3G network of an Italian telecom provider.

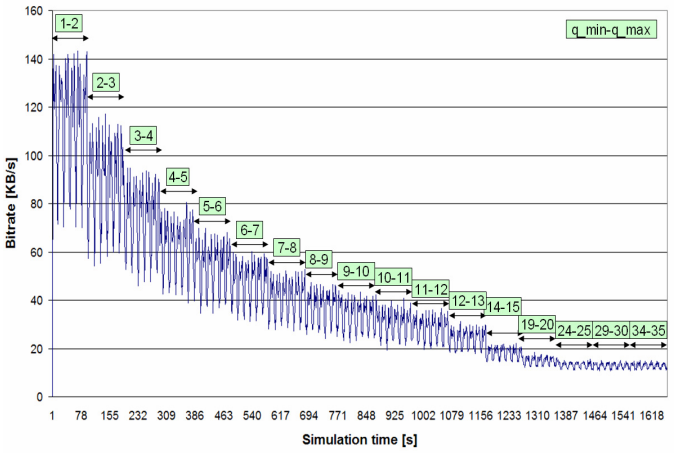

Fig. 6. Bandwidth measurements by varying quantizer values (qmin, qmax) during the same live streaming session, keeping fixed the quantization offset.

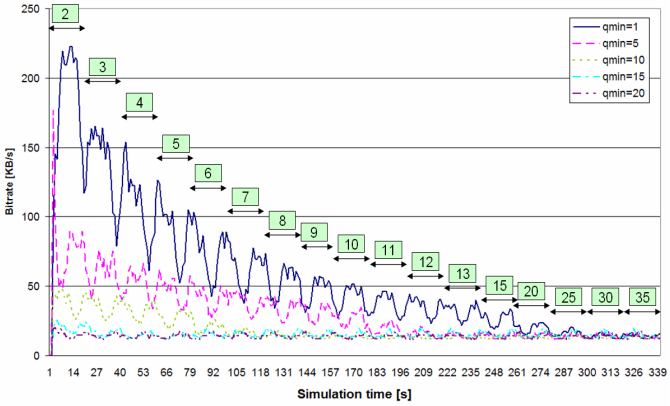

Fig. 7. Bandwidth measurements by varying both quantizer values (qmin, qmax) and the quantization offset

The use of the interactive live streaming system under analysis, in conjunction with the 802.11g standard, yields to not noticeable round-trip time values for the M-JPEG encoding. The impression of the user is that the 3D rendering is locally computed directly on the mobile device. As indicated in Table I, the round-trip time quadruples when a 3G network is used; nevertheless, the system is yet quite interactive because of the low value of the round-trip time. The image resolution during these tests was always set to 320x240 pixels. The M-JPEG image quality was set to 80. The achieved frame rate was, on the average, 15 fps. On the other hand, as shown in Table II, the round trip time becomes 10 times greater for the MPEG coding both for 802.11g and HSDPA connections. The quantizer values during this test were set to $q_{min}=1$, $q_{max}=2$.

TABLE I
M-JPEG ROUND TRIP TIME COMPARISON

Network	Average RTT
802.11g	59 ms
HSDPA 3.6 Mbps	214 ms

TABLE II
MPEG-4 ROUND TRIP TIME COMPARISON

Network	Average RTT
802.11g	583 ms
HSDPA 3.6 Mbps	1966 ms

The choice of M-JPEG instead of MPEG is basically due to the specific application to which the live video streaming system is devoted. If the application needs more fitting requirements in terms of interactivity, a M-JPEG encoding is more suitable. For example, let us consider a remote 3D rendering scenario where a user would navigate a 3D environment using a mobile device; if the intent of the user is only to inspect 3D models or scenes, a round-trip time equal to 2 seconds using a HSDPA connection and a MPEG-4 streaming flow (see Table II) can still be acceptable. On the other hand, when remote visualization is used in rendering systems for collaborative design, the average round-trip time experienced for the MPEG streaming can be detrimental to the collaboration feature.

IV. CONCLUSION AND FUTURE WORKS

This paper focuses on interactive live video streaming systems for mobile devices, where the speed of reaction to user commands is of primary importance. A solution for adaptively control encoding parameters based on feedback measures coming back from visualization clients has been proposed and it has been applied to M-JPEG streams. The live video streaming system has been tested also for other types of video encodings such as MPEG-4 Part 2, natively supported by a wide range of consumer handheld devices. Tests revealed that the proposed solution can be used elsewhere in another codec. On the other hand, experimental tests in a real scenario pointed out that the use of the MPEG live video streaming systems is subject to a smaller interactivity with respect to the M-JPEG encoding; as a consequence it can be used with applications that do not strictly care about latency and round-trip time.

Future works concern the design of an automatic controller for the adaptation of encoding parameters of the MPEG-4 encoding; moreover, we aim to investigate different types of automatic controllers. Finally, we plan to conduct also perceptive tests in order to evaluate the controller system not only from a quantitative point of view but also from the user point of view.

ACKNOWLEDGMENT

The authors wish to thank Fabrizio Agnoli for his help both in the software development and in setting up the testbed for the experiments.

REFERENCES

[1] Gutwin et al., “Revealing Delay in Collaborative Environments,” Proc. ACM Computer Human Interaction, pp. 503–510, 2004.

[2] Jae-Won Kim; Byeong-Doo Choi; Sang-Hee Park; Kyoung-Keun Kim; Sung-Jea Ko; "Remote control system using real-time MPEG-4 streaming technology for mobile robot" International Conference on Consumer Electronics, 2002. ICCE. 2002 Digest of Technical Papers. , 18-20 June 2002 Page(s):200 – 201

[3] Hamasaki, S.; Yakoh, T.; "Implementation and evaluation of decorators for delayed live streaming video on remote control system" Industrial Informatics, 2008. INDIN 2008. 6th IEEE International Conference on 13-16 July 2008 Page(s):1220 – 1225

[4] Zanaty, P.; Brscic, D.; Frei, Z.; "3D visualization for Intelligent Space: Time-delay compensation in a remote controlled environment" Human System Interactions, 2008 Conference on. 25-27 May 2008 Page(s):802 – 807

[5] Natarajan, Sriram; Ganz, Aura; "Efficient force feedback transmission system for tele surgery" Engineering in Medicine and Biology Society, 2008. EMBS 2008. 30th Annual International Conference of the IEEE. 20-25 Aug. 2008 Page(s):3245 – 3248

[6] Nave, I.; David, H.; Shani, A.; Tzruya, Y.; Laikari, A.; Eisert, P.; Fechteler, P.; "Games@large graphics streaming architecture" Consumer Electronics, 2008. ISCE 2008. IEEE International Symposium on. 14-16 April 2008 Page(s):1 – 4

[7] P. Eisert and P. Fechteler; "Remote Rendering of Computer Games". In Proc. Intern. Conf. on Signal Processing and Multimedia Applications (SIGMAP), Barcelona, Spain, July 2007

[8] Dan Ma, "The Business Model of Software-As-A-Service," IEEE International Conference on Services Computing (SCC 2007), 2007, pp. 701-702.

[9] Klaus Engel, Ove Sommer, Christian Ernst, Thomas Ertl. "Remote 3D Visualization using Image-Streaming Techniques," Proceedings of the International Symposium on Intelligent Multimedia and Distance Education.

[10] Bao, P.; Gourlay, D.; "A framework for remote rendering of 3-D scenes on limited mobile devices" Multimedia, IEEE Transactions on. Volume 8, Issue 2, April 2006 Page(s):382 – 389

[11] Lamberti, F.; Sanna, A.; "A Streaming-Based Solution for Remote Visualization of 3D Graphics on Mobile Devices" Visualization and Computer Graphics, IEEE Transactions on. Volume 13, Issue 2, March-April 2007 Page(s):247 – 260

[12] Cao, Xuefeng; Wan, Gang; "A Generic Solution for Remote Adaptive Streaming and Rendering of Planetary Terrain". Information Engineering and Computer Science, 2009. ICIECS 2009. International Conference on. 19-20 Dec. 2009 Page(s):1 – 4

[13] K. Engel, O. Sommer, and T. Ertl; "A framework for interactive hardware accelerated remote 3d-visualization". In Proceedings of EG/IEEE TCVF Symposium on Visualization VisSym'00, pages 167–177, 2000.

[14] Mavlankar, A.; Jeonghun Noh; Baccichet, P.; Girod, B.; “Peer-to-peer multicast live video streaming with interactive virtual pan/tilt/zoom functionality” Image Processing, 2008. ICIP 2008. 15th IEEE International Conference on. 12-15 Oct. 2008 Page(s):2296 – 2299

[15] Kurutepe, E.; Civanlar, M.R.; Tekalp, A.M.; "Client-Driven Selective Streaming of Multiview Video for Interactive 3DTV". Circuits and Systems for Video Technology, IEEE Transactions on. Volume 17, Issue 11, Nov. 2007 Page(s):1558 – 1565

[16] Bjntegaard, G.; Luthra, A.; Wiegand, T.; Sullivan G.J., "Overview of the H.264/AVC Video Coding Standard," IEEE Transactions On Circuit and Systems for Video Technology, july 2003.

[17] Nishantha, D.; Hayashida, Y.; Hayashi, T., "Application level rate adaptive motion-JPEG transmission for medical collaboration systems," Distributed Computing Systems Workshops, 2004. Proceedings. 24th International Conference on , vol., no., pp. 64-69, 23-24 March 2004.

[18] Endoh, K.; Yoshida, K.; Yakoh, T., "Low delay live video streaming system for interactive use," Industrial Informatics, 2008. INDIN 2008. 6th IEEE International Conference on , vol., no., pp.1481-1486, 13-16 July 2008.

[19] Paravati, G., Sanna, A., Lamberti, F., & Ciminiera, L. (2008). A novel approach to support quality of experience in remote visualization on mobile devices. In Eurographics 2008 short paper Proceedings (pp. 223-226).

[20] Kyung Chang Lee; Suk Lee; Man Hyung Lee, "QoS-based remote control of networked control systems via Profibus token passing protocol," Industrial Informatics, IEEE Transactions on , vol.1, no.3, pp. 183-191, Aug. 2005.

BIOGRAPHIES

Gianluca Paravati received the B.Sc. and M.Sc. degrees in electronic engineering from Politecnico di Torino, Torino, Italy, in 2005 and 2007, respectively.
He is currently a Research Assistant with the Dipartimento di Automatica e Informatica, Politecnico di Torino. His research interest include realtime image processing, target tracking in infrared imagery, collaborative virtual environments, and data distribution systems.

Cesare Celozzi received the M.Sc. degree in computer engineering from Politecnico di Torino, Torino, Italy, in 2004. He is currently a Ph.D. student with the Dipartimento di Automatica e Informatica, Politecnico di Torino. His research interest include virtual and augmented reality environments, human-machine 3D interfaces and segmentation of medical images.

Andrea Sanna received the M.Sc. degree in electronic engineering and the Ph.D. degree in computer engineering from Politecnico di Torino, Torino, Italy, in 1993 and 1997, respectively.
He is currently an Assistant Professor with the Dipartimento di Automatica e Informatica. He is the author or coauthor of several papers in the areas of computer graphics, virtual reality, parallel and distributed computing, scientific visualization, and computational geometry. He is currently involved in several national and international projects concerning grid, peer-to-peer, and distributed technologies.
Dr. Sanna is a member of the Association for Computing Machinery and serves as Reviewer for a number of international conferences and journals.

Fabrizio Lamberti (M'02) received the M.Sc. degree in computer engineering and the Ph.D. degree in software engineering from Politecnico di Torino, Torino, Italy, in 2000 and 2005, respectively.
He is currently with the Dipartimento di Automatica e Informatica, Politecnico di Torino. He is the author of a number of technical papers published in international journal and conference proceedings in the areas of mobile and distributed computing, wireless networking
Dr. Lamberti is member of the IEEE Computer Society. He is also a member of editorial advisory boards of international journals.