Online QoS Management for Multimedia Real-Time Transmission in Industrial Networks

Javier Silvestre-Blanes, *Member, IEEE*, L. Almeida, *Member, IEEE*, R. Marau, *Member, IEEE*, and P. Pedreiras, *Member, IEEE*

***Abstract*—A growing number of industrial applications incorporate multimedia information processing. These multimedia applications are commonly distributed and subject to time constraints that must be met across networks without creating intolerable interference over typical control flows. However, multimedia traffic, in general, and video streaming, in particular, have specific characteristics that conflict with the operational framework of conventional real-time protocols. In particular, video compressors generate highly variable bit-rate streams that mismatch the constant-bit-rate channels typically provided by real-time protocols, severely reducing the efficiency of network utilization. This paper focuses on low-latency multimedia transmission over Ethernet with dynamic quality-of-service (QoS) management. We propose a multidimensional mechanism that controls, in an integrated way, both the compression parameters and the network bandwidth allocated to each stream. The goal is to provide the best possible QoS to each stream, recomputing the compression levels and network bandwidth whenever significant events, such as channel setup/teardown, or structural changes happen. This paper also presents novel QoS metrics based both on the image quality and network parameters. Several experiments with prerecorded video streams illustrate the advantages of the proposed approach and the convenience of the metrics.**

***Index Terms*—Industrial networks, multimedia, quality-of-service (QoS) management.**

I. Introduction

THE dissemination of media control applications (MCAs) [1] such as machine vision [2], automated inspection [3], object tracking [4], and vehicle guidance [5]–[7] in industry is increasing strongly. Industrial MCA can be classified into two broad classes [1], namely, *supervised multimedia control subsystems* [8] and *multimedia embedded systems* (MESs). In the first application class, the emphasis is essentially on the quality of the media processing, while the real-time constraints are essentially soft. The latter class of MCA is more demanding since, in addition to the media processing quality, hard real-time requirements also come into play. Typically, these applications are complex and heterogeneous, encompassing several real-time activities in addition to the media processing ones. Thus, the interference caused to and suffered by the multimedia-handling components must be limited and predictable.

Many of the MES applications are distributed, relying on real-time network protocols that provide the necessary real-time communication services. However, multimedia traffic, in general, and video streaming, in particular, have specific characteristics that conflict with the operational framework of conventional real-time protocols. In particular, multimedia information is, due to the compressors used, a variable bit-rate (VBR) traffic source, whereas the real-time networks usually offer to the application constant-bit-rate (CBR) channels (e.g., PROFINET-IRT, ATM, ControlNet, Interbus, or flexible time-triggered (FTT) networks [9]–[12]). Matching a VBR source to a CBR channel is not trivial and may lead either to a waste of bandwidth or rejection of frames. This difficulty became particularly challenging with the emergence of MES applications, described earlier, which typically impose reliability and timeliness requirements that cannot be fulfilled by standard network protocols [13], due to a lack of temporal isolation and consequent unbounded mutual interference between streams.

In a previous work [14], the authors proposed taking advantage of the dynamic quality-of-service (QoS) management features of the FTT communication over switched Ethernet (FTT-SE) protocol to perform the VBR-to-CBR adaptation with MJPEG video streams. This adaptation is based on a multidimensional mechanism that manages, in an integrated way, the compression parameters and the network bandwidth allocated to each stream. The streams' bandwidth is adjusted online, depending on their relative importance, current compression levels, and global network utilization. The goal is to provide, at every instant, the best possible QoS to each stream, recomputing the compression levels and the allocated network bandwidth in response to significant events such as channel setup/teardown or video structural changes.

This paper extends the work presented in [14] into four main aspects. First, in addition to the channel period, the QoS layer can now also adapt the channel width (C), thus enlarging the configuration space. This approach brings higher flexibility and increased granularity to the channel bandwidth allocation mechanism. This feature is of high practical relevance, since most of the imaging devices restrict the frame acquisition periods to discrete predefined sets, resulting in a correspondingly discrete stepwise bandwidth allocation function that limits the

Manuscript received August 14, 2009; revised November 2, 2009 and February 27, 2010; accepted April 9, 2010. Date of publication May 10, 2010; date of current version February 11, 2011. This work was supported in part by the Ministery of Science and Technology of Spain under Project TSI2007-66637-C02-02, by the iLAND Project (call 2008-1 of the European Union ARTEMIS Joint Undertaking Program), by the European Community through the ICT NoE 214373 ArtistDesign, and by the Portuguese Government through the FCT Project HaRTES (PTDC/EEA-ACR/73307/2006).

J. Silvestre-Blanes is with the Instituto Tecnológico de Informática, Universidad Politécnica de Valencia, 46022 Valencia, Spain (e-mail: jsilves@disca.upv.es).

L. Almeida and R. Marau are with the University of Porto, 4099-002 Porto, Portugal (e-mail: lda@fe.up.pt; marau@fe.up.pt).

P. Pedreiras is with the University of Aveiro, 3800 Aveiro, Portugal (e-mail: pbrp@ua.pt).

Digital Object Identifier 10.1109/TIE.2010.2049711

configuration space options. Therefore, adjusting the channel width and period together allows obtaining smooth mode changes and provides a much richer configuration space. In addition, for some systems, changing the channel period is undesirable due to rate coupling among different channels or to constraints imposed by application-level controllers (e.g., sample period). In this case, the only possibility is to adapt the channel width while keeping the respective period constant.

Second, this paper also extends the work presented in [14] by using a richer set of inputs to the QoS manager. In particular, the computation of the system benefit takes into account the image quality, as well as the bandwidth usage, allowing different benefit/cost tradeoffs. This paper also extends significantly the experimental results included in [14], by including tests using more streams, with more dynamic requirements, in more scenarios (five dynamic and four static experiments) and for longer duration. Finally, the description of the state of the art has also been significantly enhanced.

This paper is organized as follows. Section II presents an overview of the related works. Section III presents the system model considered in this work. An experimental section follows in Section IV, presenting a set of experiments carried out to assess the performance of the methodology proposed. Conclusions are drawn in Section V.

II. Related Works

A. Multimedia Compression Standards

Multimedia compressors attempt to identify redundant data, e.g., groups of pixels of similar color, to reduce the data size. Depending on the principle of operation, multimedia compression standards can be divided into two main classes: still-image compressors and video compressors. Still-image compressors use intraframe compression (i.e., the data being compressed are exclusively within the frame), while video compressors exploit the temporal redundancy that exists in sequentially acquired images. The most common still-image compressor standards are the JPEG [15] and, more recently, JPEG2000 [16]. With respect to video compressors, the most common standards are MPEG-2 [17], H.263 [18], and MPEG-4 part 2 [19] and part 10 [20], also known as H.264 or MPEG-4 AVC.

Selecting the most adequate compression technique is not a trivial matter and, to some extent, is application dependent. Video coding normally results in higher compression rates and, hence, lower bandwidth requirements than still-image coding. On the other hand, in video coding, the reaction to network load variation affects the compression level and, thus, the image quality, while the frame rate is kept constant. This approach is adequate for monitoring applications but not for MES or to surveillance/recording applications [21] which are often found in industrial environments. Furthermore, still-image transmission is more robust than video transmission. This conclusion can be drawn from the fact that, in still-image compression, the frames are independent of each other, and thus, losing one image or parts of it has no consequence for the following images. In turn, video transmission uses different frame types, namely, I-frames, which are independent, but also P-frames, i.e., interframes coded depending on previous frames, and sometimes B-frames that depend on following frames. Only I-frames are self-contained; thus, the loss of a frame or part of it may have an impact on several of the following frames, until the arrival of another I-frame. This effect is further aggravated by a common practice that consists in enlarging the distance between I-frames to reduce the bandwidth utilization. Another aspect that penalizes the use of video compression in industrial applications is that the images of different streams are sometimes captured at low rates and multiplexed together in the same channel. Any of these situations can reduce severely the temporal redundancy and, thus, the level of compression that can be attained.

Whenever timeliness requirements come into play, the lower latency of JPEG with regard to other video/still-image compression techniques presents a significant advantage. The relevance of this aspect is growing in consequence of the use of increasingly higher resolution image sensors, which have evolved from the traditional Common Intermediate Format (CIF) and Quarter CIF (352×288 and 176×144 pixels), with a maximum size of 4CIF 704×576 pixels, to Video Graphics Array with 640×480 pixels, Extended Graphics Array (XGA) with 1024×768 pixels, Super XGA with 1280×1024 pixels, and Widescreen Ultra XGA with 1920×1200 pixels, arriving to the Wide Hexadecatuple Ultra XGA format, with 7680×4800 pixels. Thus, the amount of data to be processed in each frame is growing exponentially, emphasizing the importance of the compressor latency.

B. Multimedia Transmission

Multimedia transmission over the Internet has been the subject of intense research in recent years [22]–[24]. Typical solutions are based on the Transmission Control Protocol/User Datagram Protocol/Internet Protocol (IP) protocol stack complemented by other protocols, e.g., Real-Time Protocol/Real-Time Control Protocol [25], Real-Time Streaming Protocol [26], or Session Initiation Protocol [27], which measure key network parameters, such as bandwidth usage, packet loss rate, and round-trip delays to control the load submitted to the network.

The main drawback of these technologies concerning their use for industrial communications is the latency introduced. For example, smoothing video algorithms [28] use memory buffers between the producer and the consumer to smooth out the bit-rate variations. The estimation of the required bandwidth and of the amount of buffering can be done offline, for stored video, or based on a number of images buffered before their transmission, for live, i.e., noninteractive, video streams. The quality of results is, however, highly dependent on the delay allowed by the application. Similar limitations are found in the content-based network resource allocation schemes presented in [29] and [30]. In these approaches, the latency can be very high since they employ relatively large image buffers in the sender and are based on standard IP networks, using traffic smoothing techniques. Furthermore, they require a complementary processing stage before compression, in order to adapt the compression to the image content, thus further increasing the latency and incurring in high computational overhead in the side of the sender nodes. The latency problem is addressed by

the low-delay rate control algorithms [31]–[33], such as those used in TMN8 [34], which achieve a high level of performance for the applications for which they have been designed, mainly videophone and videoconference. However, these algorithms are based on the use of only one I-frame and a following sequence of P-frames, an approach that can be used in soft real-time applications like videophone and videoconference, but not in MES applications. For example, the appearance of a new object in the scene, which has to be processed by a computer vision system in order to take a decision affecting the industrial process, will either generate a traffic peak, if the compression is kept unchanged, or a strong quality reduction, if the compression is changed to keep the bandwidth utilization stable. Any of these situations is a potential source of perturbations that may negatively affect MES applications, despite the efficiency and flexibility of those protocols with generic video transmission.

The overall latency in video transmission arises from two main components, namely, the video codecs and the network delay/losses. The former ones can be strongly reduced by using still-image coding, as referred before. The latter ones can be improved by using real-time communication protocols, which usually provide CBR channels with bounded latency, as referred to in Section I. Their use, however, requires matching the VBR streams generated by the video encoders to the fixed bandwidth provided by the communication channels. This can be done by adapting some parameters [35], [36] like image resolution or, more frequently, changing the frame rate, which implies dropping frames, with both of them impacting at the application level. There are different approaches to this matching. Taking a conservative approach, one could reserve a channel with a capacity equal to the maximum bandwidth required by the multimedia source. While taking this approach guarantees that no frames are lost, the fact that the bandwidth requirement generated by multimedia sources typically exhibits a high variance leads to a potentially significant bandwidth waste. One possible approach to overcome this inefficiency problem would be reserving a channel with a capacity equal to the average required bandwidth. Despite being more efficient from a bandwidth point of view, this approach can lead to additional delays or to frame losses, depending on the existence and size of buffers at the source nodes, whenever the instantaneous required bandwidth exceeds the average value. Moreover, note that many applications comprise the transmission of several multimedia streams, thus multiplying the impact of these sources of inefficiency.

The difficulty in fitting VBR sources in CBR channels motivated the work presented in this paper. The dynamic QoS management features of the FTT-SE protocol are used to adapt dynamically the bandwidth of the real-time communication channels. The adaptation mechanism takes as inputs the relative importance, the current allocated bandwidth, and the current compression level of each multimedia source, as well as the global network utilization, recomputing the compression levels and the allocated network bandwidth in response to significant events such as channel setup/teardown or video structural changes. The goal is to provide, at every instant, the best possible QoS to each stream. The admission control and scheduling capabilities of the FTT-SE protocol allow carrying out such changes online with real-time guarantees [37], thus being suitable for MES applications. This situation is rather different from video transmission over the Internet, as referred earlier in this section, in which case there is a very limited, if any, control over the communication channel and, particularly, there are no separate channels with guaranteed and isolated bandwidth for each stream.

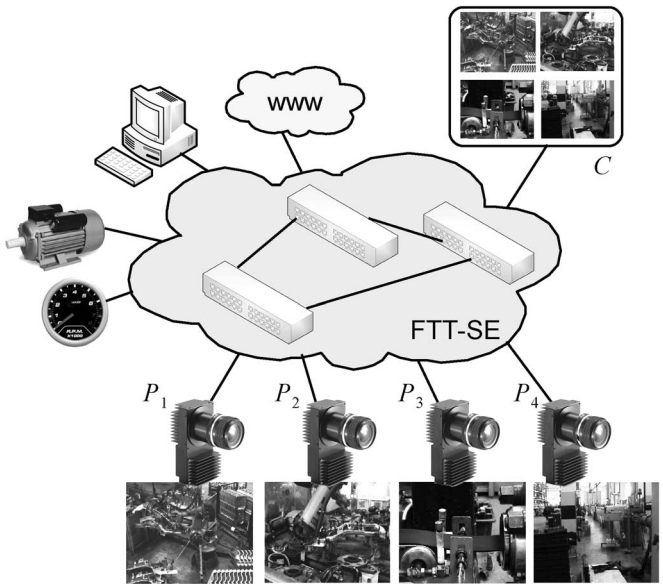

Fig. 1. System architecture.

III. System and QoS Models

In this paper, we consider p multimedia sources, also called *producers*, that send a set $\mathbb{M} \equiv \{M_i, i = 1, \ldots, p\}$ of video streams to c multimedia sinks, called *consumers*, via a local area network. Aside from the video streams, the network may also support other traffic sources, potentially with stringent real-time requirements, e.g., related to real-time control, as well as nonreal-time sources, e.g., related to configuration or even remote access over the Internet for maintenance purposes (see Fig. 1).

The FTT-SE protocol was selected to address these communication requirements. This is a real-time master–slave protocol, that includes features particularly well suited for supporting the needs of the framework herein presented, namely, dynamic traffic scheduling, online admission control, dynamic QoS management, and support of both isochronous and asynchronous traffic with temporal isolation [38]. FTT-SE networks comprise a master node, which holds the message properties, a scheduler, an admission control block, and a QoS manager. Slave nodes implement a transmission control layer that mediates the access to the network. The master node periodically broadcasts a control message (trigger message) that contains the IDs of the messages that should be transmitted within a predefined interval, designated elementary cycle (EC). The master schedules the traffic dynamically, once every EC; thus, change requests are promptly reflected at the network level. The FTT-SE protocol reserves part of the EC for real-time traffic, enforcing mutual isolation between traffic classes. This latter

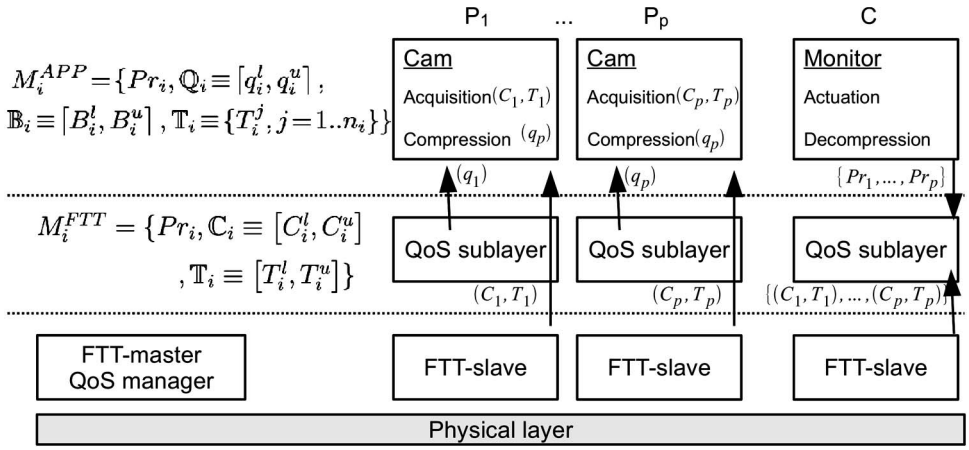

Fig. 2. QoS management model.

feature allows focusing on the multimedia traffic, only, for the remainder of this paper, since it will be handled in isolation.

In the remainder of this paper, we consider a scenario in which several nodes send video streams to a sink, with a single producer per transmitting node. Additionally, all nodes reside in a local area network, and thus, the communication between the sources and the sink is direct. This scenario is often found in MES applications, such as automated inspection, in which a set of sensor nodes sends video data to a controller node or operator console that has decision capacity. Nevertheless, this is, by no means, a restriction of the proposed system, which equally supports more complex scenarios such as with many-to-many stream transmissions. At the application level, the QoS model (Fig. 2) considers each stream being characterized by $M_i^{\mathrm{APP}} = \{Pr_i, \mathbb{Q}_i, \mathbb{B}_i, \mathbb{P}_i\}^{\mathrm{APP}}$, where Pr_i^{APP} is the stream normalized relative priority $(\sum_{\forall_i} Pr_i = 1)$, $\mathbb{Q}_i^{\mathrm{APP}} \equiv [q_i^l, q_i^u]$ is the range of allowed quantification factors that imply different compression levels, $\mathbb{B}_i^{\mathrm{APP}} \equiv [B_i^l, B_i^u]$ is the range of possible frame sizes after compression, and $\mathbb{P}_i^{\mathrm{APP}} \equiv \{P_i^j, j = 1, \ldots, n_i\}$ is the set of possible interframe intervals corresponding to the n_i allowed frame rates.

The QoS manager receives channel setup, teardown, and change requests from different system nodes and assigns bandwidth to each channel following a QoS-based criterion. This FTT QoS management interface considers each channel characterized by $M_i^{\mathrm{FTT}} = \{Pr_i, \mathbb{C}_i, \mathbb{T}_i\}^{\mathrm{FTT}}$, where Pr_i^{FTT} is a priority that reflects the relative channel importance, $\mathbb{C}_i^{\mathrm{FTT}} \equiv [C_i^l, C_i^u]$ is the range of possible transmission buffer sizes, and $\mathbb{T}_i^{\mathrm{FTT}} \equiv [T_i^l, T_i^u]$ is the range of possible transmission periods. The output of the QoS manager is the actual bandwidth w_i assigned to each channel at each instant and materialized as a (C_i, T_i) duplet that is communicated back to the QoS sublayer and application. Note that $w_i = C_i/T_i$.

A. QoS Sublayer

Within the nodes, the QoS sublayer is responsible for mapping the application QoS parameters onto network QoS parameters. This mapping is straightforward with $Pr_i^{\mathrm{FTT}} = Pr_i^{\mathrm{APP}}$, $\mathbb{T}_i^{\mathrm{FTT}} = \mathbb{P}_i^{\mathrm{APP}}$, and $C_i^l = B_i^l$. The upper bound of the transmission buffer size range C_i^u is determined in each QoS negotiation, expressing the desired buffer size from the QoS sublayer perspective at that moment, as detailed in Section III-B.

Each QoS sublayer is also responsible for fitting the multimedia encoded stream within the granted channel bandwidth w_i. This is achieved by adapting the quantification level q_i, possibly discarding frames, and even renegotiating the channel bandwidth with the QoS manager, whenever appropriate. The adaptation of q is based on the $R(q)$ frame bandwidth model [39], where α and λ are considered constant for each stream, β is the frame specific, and $\overline{q} = 100 - q$ is the compression level which varies symmetrically with respect to the quantification factor

$$R(q) = \alpha + \frac{\beta}{\overline{q}^{\lambda}}. \tag{1}$$

The actual use of this $R(q)$ model is detailed in [14], and it allows deriving at instance k an estimate of the quantification level for the next frame q_i^{k+1} that will generate a bandwidth R_i^{k+1} within a *channel target window*. As long as the frame bandwidth falls inside such a window, q is kept, and its adaptation is not invoked, thus reducing the frequency of adaptations and saving overhead. This window is controlled by a parameter δ, resulting in $[w_i(1-3\delta), w_i(1-\delta)] \equiv [R_i^T \pm w_i\delta]$. The value of q_i^{k+1} is then defined as a function of the current frame bandwidth R_i^k, the current channel bandwidth w_i, and δ.

Fig. 3 shows three possible scenarios at time t^{k+1} in which the resulting frame bandwidth falls outside the channel target window. In scenario a), the generated frame bandwidth R_i^{k+1} exceeds the current channel width w_i, causing a frame drop; in b), it is within the channel width but over the target window, while in c), it is below the target window, leading to an underutilization of the channel bandwidth. In all three scenarios, the adaptation is invoked to compute an estimate of the quantification level q_i^{k+2} that will generate an R_i^{k+2} that falls within the channel target window.

The δ factor is a predefined relative fraction of the channel bandwidth, equal for all channels, that sets a compromise between higher efficiency in channel bandwidth utilization (lower δ values) and lower frequency of compression level adaptation invocations (higher δ values). Currently, δ is set empirically.

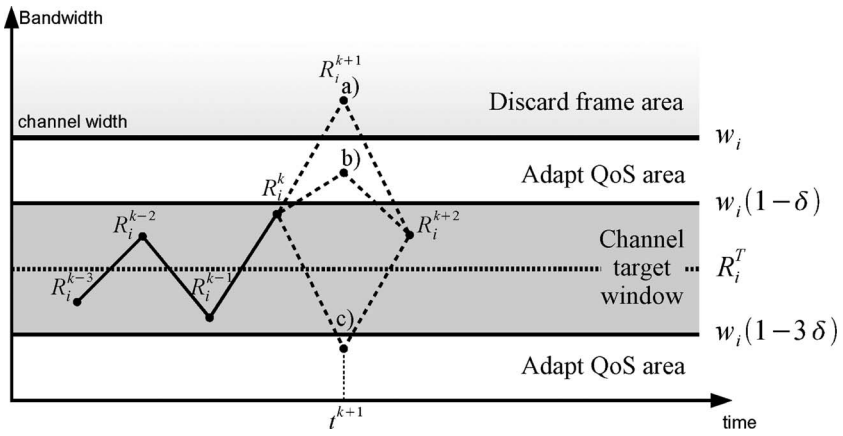

Fig. 3. Fitting a VBR stream into a CBR channel.

Future work will consider the use of channel-specific δ parameters and will analyze their optimization.

B. QoS Manager

The QoS manager distributes the network link bandwidth, referred to as U^S, among the channels according to a predefined QoS policy, the channel QoS parameters, and according to the current number of active channels. Note that U^S is a bandwidth bound that assures the timeliness of the communication channels in each link according to the scheduling policy in use, e.g., based on the rate monotonic or earliest deadline first scheduling bounds. This bandwidth distribution occurs within the channel renegotiation procedure that is triggered sporadically as a response to the online connection/disconnection of streams, to explicit requests from an operator to maximize the QoS of a given stream, or to significant structural changes in any of the active streams. In the first two cases, the bandwidth redistribution is triggered externally, while in the latter case, it is triggered autonomously by a sequence of frames that are dropped or that fall within the channel but above or below the target window. In order to detect these situations, two counters are used, namely, the $over\_cnt$ and $under\_cnt$ counters, which count the number of consecutive frames that fall above and below the target window. A QoS renegotiation is autonomously triggered whenever any of these counters exceeds a predefined *quality change threshold* QCT. This threshold is a system parameter that controls the frequency of autonomous channel renegotiations. The higher this parameter is, the longer it will take for the system to trigger a channel renegotiation.

Algorithm III.1 shows the hierarchy of procedures involved in the QoS adaptation process. First, the QoS sublayer autonomously adjusts the quantification factor, in a frame-by-frame basis, trying to keep the stream bandwidth within the target window. If the quantification factor falls out of the allowed range, the system uses the nearest valid quantification factor (saturation function). Eventually, the frame may be dropped since the simple adaptation of the quantification was not enough to bring the stream bandwidth to the channel target window. Whenever this scenario repeats more than QCT consecutive times, it is considered that a long-lasting situation is happening, resulting, for example, from structural changes in the image or from an explicit QoS parameter change made by the application, and consequently, a QoS renegotiation is requested. Otherwise, the overload is considered as spurious and handled locally by the QoS sublayer without impact on the communication channels.

Algorithm III.1: $q_i' = qosAdaptation(R_i, q_i)$

---

comment: computes q_i for next frame inside producer M_i
comment: R^{inv} is the inverse model of $R(q)$
$q_i' \leftarrow q_i$
if $R_i > w_i(1-\delta)$
  then $\begin{cases} q_i' \leftarrow R^{\text{inv}}(R_i^T) \\ over\_cnt \leftarrow over\_cnt + 1 \end{cases}$
  else $over\_cnt \leftarrow 0$
if $R_i < w_i(3-\delta)$
  then $\begin{cases} q_i' \leftarrow R^{\text{inv}}(R_i^T) \\ under\_cnt \leftarrow under\_cnt + 1 \end{cases}$
  else $under\_cnt \leftarrow 0$
if q_i' **not in** $[q_i^l, q_i^u]$
  then $\begin{cases} q_i' \leftarrow saturation(q_i^l, q_i^u) \\ \textbf{if } over\_cnt > QCT \textbf{ or } under\_cnt > QCT \\ \quad \textbf{then } qosRenegotiation() \end{cases}$
return (q_i')

Whenever a QoS renegotiation ($qosRenegotiation()$ in Algorithm III.1) is requested, several steps have to be performed. First, the desired bandwidth for each stream w_i^d is computed. Then, if the link bandwidth U^s is not enough to satisfy the desired values, a bandwidth distribution algorithm is used to compute the effective bandwidth (w_i) that each stream is allowed to use (Algorithm III.2). Finally, the computed bandwidth of each stream has to be translated into FTT operational parameters (C, T) (Algorithm III.3). Note that there are many different possibilities to carry out both the bandwidth distribution and the mapping of stream bandwidth onto network parameters. The algorithms presented here and explained next are just one possibility that, nevertheless, is effective. Performing an extensive analysis and comparison of different algorithms for these purposes is out of the scope of this paper.

Algorithm III.2: $\mathbb{W}^{\text{qos}} = wDistribution(\mathbb{M}, U^S)$

comment: distributes the system bandwidth capacity
$U_{\text{spare}} \leftarrow U^S - \sum_{\forall i} w_i^{\min}$
for each $M_i \in \mathbb{M}, sorted\ by\ Pr_i$
do
- **if** $(w_i^d - w_i^{\min}) < U_{\text{spare}}$
  - **then** $w_i' \leftarrow (w_i^d - w_i^{\min})$
  - **else** $w_i' \leftarrow U_{\text{spare}}$
- $U_{\text{spare}} \leftarrow U_{\text{spare}} - w_i'$
- $w_i^{\text{qos}} \leftarrow w_i^{\min} + w_i'$

return $(\mathbb{W}^{\text{qos}} = \{w_1^{\text{qos}}, \ldots, w_n^{\text{qos}}\})$

Algorithm III.3: $\{(C_i, T_i)\forall_i\} = uCTmapping(\mathbb{W}^{\text{qos}}, \mathbb{M})$

comment: $succ(T_i)$ is the successor of T_i in the monotonically increasing set $\mathbb{T}_i^{\text{FTT}}$
for each $M_i \in \mathbb{M}$
do
- $T_i = \max\{T_i^j, \forall_{j=1,\ldots,n_i} : T_i^j \leq C_i^u / w_i^{\text{qos}}\}$
- $C_i = w_i^{\text{qos}} * T_i$
- **if** $C_i < C_i^l$
  - **then**
    - $T_i = succ(T_i)$
    - $C_i = C_i^u$
- $w_i = C_i / T_i$

return $((C_i, T_i)\forall_i, \mathbb{W} = \{w_1, \ldots, w_n\})$

Finally, note that Algorithm III.1 (excluding the QoS renegotiation) that executes in the end nodes, as well as both Algorithms III.2 and III.3 that execute in the master node, incurs on a negligible computation overhead that, in a common PC hardware, may represent, at most, a few microseconds. Thus, this can be done without problems on a per-frame basis. On the other hand, a QoS renegotiation request implies a set of FTT-related actions, including a request from one slave to the master, to trigger the process and the communication of the new parameters by the master back to the slaves. In the worst case, the first operation may take two ECs, and the second one may take one EC. In the current setup, the EC was 1 ms long; thus, the total communication latency is, at most, 3 ms. Comparing with the frame periods, this is still a relatively small latency that has no practical impact on the video process. Thus, the current system would even withstand a QoS renegotiation every frame, if needed. Nevertheless, the overhead control mechanisms included in this framework are still valuable since they allow using longer ECs and low computing power platforms, increasing the configuration flexibility.

1) Distributing Bandwidth Among Channels: When a node needs to carry out a QoS renegotiation, its QoS sublayer starts by estimating the new desired transmission buffer sizes C_i^u. These are determined in a way to fulfill the maximum bandwidth requirement of each stream at each instant, i.e., using the $R(q)$ model with the highest quantification level (minimum compression), considering the current interframe interval T_i. Such values are then capped to the application-specified upper bound on the frame size B_i^u. The following expression shows how these values are computed:

$$C_i^u = \min\left(B_i^u, \frac{R(q_i^u)}{(1-2\delta)} \times T_i\right). \tag{2}$$

Once the desired channel bandwidths are determined, the QoS sublayer hands them over to the FTT QoS manager through the FTT QoS interface. The first operation of the QoS manager is to compute, for each channel, the desired bandwidth $(w_i^d = C_i^u/T_i)$, as well as the minimum bandwidth $(w_i^{\min} = C_i^l/\max(T_i^j)\ \forall j = 1, \ldots, n_i)$. The minimum bandwidth is always checked upon addition of a new stream as part of an admission control that is embedded in the QoS management. In fact, a stream can only be accepted if all the minimum channel bandwidths, including its own, can be granted.

The bandwidth distribution algorithm is arbitrary. Different policies can be seamlessly used within the QoS manager inside the FTT master, without requiring any further changes in the rest of the system. For instance, the FTT-SE protocol was used in the context of the FRESCOR contract-based framework to deal with the network layer [43].

Algorithm III.2 shows a fixed priority-based policy that was implemented in this work. It starts from the minimum bandwidth requirements $(w_i^{\min})$ and distributes the remaining bandwidth among the channels following a strict priority order according to the Pr_i parameter and until there is no more system bandwidth to assign. In most cases, there will not be enough system bandwidth to satisfy all channel requests. In such circumstance, some channels will get the requested bandwidth; others will just get their minimum requirement bandwidth, while others will get an intermediate value of bandwidth between the previous two cases.

2) Mapping Bandwidth Onto Actual Channel Parameters: The result of the bandwidth distribution in the previous step is the set of channel bandwidth assignments (w_i^{qos}) for all channels in the system. However, the bandwidth itself is not an operational parameter. Consequently, it must be converted into a (C_i, T_i) duplet to be used by the QoS and FTT sublayers. This conversion is not univocal since different (C_i^j, T_i^j) pairs may produce the same bandwidth. Furthermore, at the FTT level, C is bounded, and T may have restrictions, e.g., due to the need to match camera frame-rate restrictions, thus, a direct correspondence between w_i^{qos} and a (C, T) pair may or may not exist. In this case, the mapping algorithm has to compute a bandwidth value that approaches, without exceeding, the bandwidth granted by the bandwidth distribution algorithm (i.e., $w_i = (C_i/T_i) \leq w_i^{\text{qos}}$). Several mapping approaches are possible, and choosing the best one is application dependent. Algorithm III.3 describes a mapping approach that attempts to maximize the transmitting budget C_i, bringing it as close as possible to the application desired upper value C_i^u while keeping the allocated channel bandwidth w_i. To do so, first, the algorithm computes the period T that corresponds to the allocated bandwidth w_i^{qos} with $C_i = C_i^u$. Since the periods are discrete, we use the closest but lower value in the monotonically increasing set $\mathbb{T}_i^{\text{FTT}}$. This

TABLE I
STREAM PROPERTIES FOR EXPERIMENTS D1–D4

d 1-4 [14]	M_1 CF3/CF1	M_2 RB2/RB1	M_3 RB2	M_4 RB1	M_5 CF3	M_6 CF1
q_l	20	40	40	20	30	15
q_u	70	70	70	70	50	55
T_l(ms)	40	40	40	40	40	40
T_u(ms)	160	120	160	120	120	120
B_l(B)	30k	30k	30k	30k	20k	25k
B_u(B)	50k	50k	50k	50k	60k	55k
Pr	0.166	0.166	0.166	0.166	0.166	0.166

approximation eventually leads to a bandwidth w_i that can be greater than the allocated one. In such cases, C_i is recomputed to match the allocated bandwidth. However, in the sequel, it may happen that the computed C_i violates the defined lower bound $(C_i < C_i^l)$. In that case, the next value in the period list $succ(T_i)$ is selected, and C_i is made equal to C_i^u, which means that an exact bandwidth match cannot be found resulting in a reduced bandwidth w_i. Finally, note that, as long as $w_i \geq w_i^{\min} = C_i^l/T_i^u$, $C_i < C_i^l$ implies that $T_i < T_i^u$, and thus, there will always be $succ(T_i)$ in the set in that case.

IV. EXPERIMENTAL RESULTS

In order to assess the performance of the multidimensional content scaling technique presented in this paper, several experiments were carried out using the streams described in Table I. This set of streams is based on those presented in [40], consisting of prerecorded sequences obtained in industrial environments, namely, car factory and rubber factory environments. Streams M_1 and M_2 have been introduced with the objective of testing more dynamic scenarios. Both streams are obtained alternating frames of streams M_5 and M_6 in the first case and M_3 and M_4 in the second case, representing a situation in which different video sources are multiplexed. Stream M_3 is representative of industrial surveillance applications, showing frequent changes in the bandwidth requirements. Stream M_4 presents a more static scenario, with nearly constant requirements. Streams M_5 and M_6 present smooth variations alternated with strong peaks, representing scenarios with sudden changes in the environment such as those caused by sparks in welding machines. Fig. 4 shows the evolution of the frame size in each stream compressed with a constant q, illustrating its dynamics, complexity, and requirements.

A. Experiment Characterization

A total of five different dynamic experiments were carried out. The first group of experiments, denoted d1–d4, was designed to assess the influence of the δ and QCT parameters. Experiment d5 allowed us to assess the impact of the QoS priority. Tables I and II depict the QoS parameters of each experiment. For experiments d1–d4, the duplet (δ, QCT) takes the values (0.1, 1), (0.05, 1), (0.1, 2), and (0.05, 2), respectively, while the stream priority is kept equal for all streams. Conversely, in experiment d5, the duplet (δ, QCT) is equal to experiment d2, while the streams receive different priorities, as shown in the last line in Table II. To establish a baseline for the performance gains, a set of static experiments with fixed q, C, and T was also carried out. Experiments s1–s3 use $C = 44$ kB and $T = 80$ ms, resulting in a total bandwidth of 26.4 Mb/s, with quantification factors set to 50, 55, and 60, respectively. In experiment s4, C was set at 22 kB, and T was set to 40 ms, yielding a similar bandwidth, while the quantification factor q was set to 15.

B. Global QoS Metrics

In order to assess the impact of the QoS management techniques, we use several global QoS metrics that compare the received streams with the raw original ones, frame by frame. In particular, the quality of the received images is assessed with the classic peak signal-to-noise ratio [(PSNR); measured in decibels], as well as with the quality index (QI) [41], which is believed to provide a better correlation with human perception than the PSNR.

Video quality is usually calculated using the frame quality average, sometimes weighted depending on some stream properties [42]. Usually, these metrics consider average image degradation only, ignoring the efficiency of the channel bandwidth utilization. Herein, we propose a new metric that weights each stream with its priority and accounts for the efficient use of the channel bandwidth by favoring the streams that present lower bandwidth waste. The formula is the following, where n is the number of frames and Wb_i is the wasted bandwidth in stream M_i

$$QoS_i' = \frac{Pr_i}{1 + Wb_i} \sum_{k=1}^{n} QI_i^k. \tag{3}$$

The global QoS' is also computed as the average of the QoS_i' parameters. In the following experiments, we will use QI and PSNR to characterize the quality of each individual stream and the QoS' metric for assessing the aggregated QoS of each experiment.

C. Results

Tables III and IV report the experimental results obtained. For each experiment and video sequence, the tables show the number of dropped frames (DrF), the wasted bandwidth Wb (measured in megabits per second), and the quality according with the PSNR and QI criteria.

Table III shows that the parameter δ has a noticeable effect on the system's behavior. Reducing δ causes a consistent reduction on the wasted bandwidth, as expected. However, this reduction is achieved at the expense of an increased number of dropped frames. This effect is particularly visible in streams that exhibit higher dynamics (e.g., M_5), while for streams that have more stable requirements, the impact is minor or even null (e.g., M_3). The impact on the number of dropped frames is, however, not always reflected in the image quality metrics. The justification for this fact is that making δ narrower increases the number of dropped frames but, at the same time, also raises the target bandwidth window, leading to a higher efficiency in using the channel width, allowing the QoS adaptation layer to use higher quantification (low compression) values. In most of the streams, the increase in quality compensates the higher number of dropped frames, with the final difference not being statistically

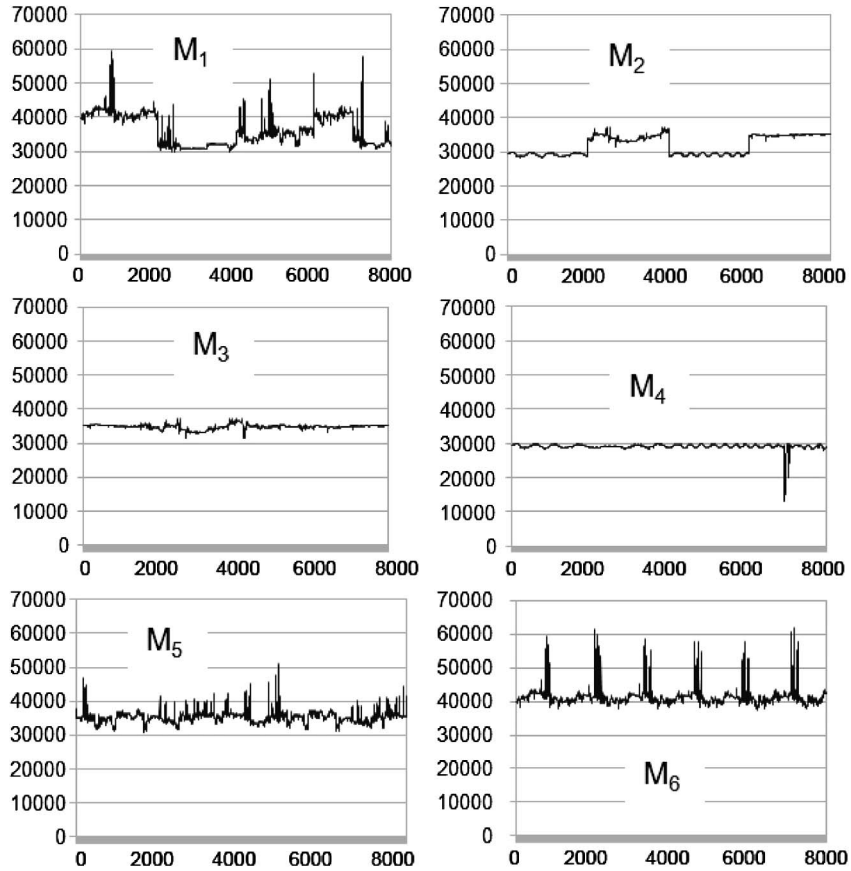

Fig. 4. Frame size evolution for a constant $q = 55$.

TABLE II
STREAM PROPERTIES FOR EXPERIMENT D5

d5	M_1	M_2	M_3	M_4	M_5	M_6
q_l	30	30	30	20	30	20
q_u	70	50	50	40	70	70
T_l(ms)	40	80	80	80	40	40
T_u(ms)	80	160	160	200	80	120
B_l(B)	30k	20k	20k	30k	20k	25k
B_u(B)	60k	60k	60k	60k	70k	65k
Pr	0.25	0.10	0.10	0.10	0.25	0.20

significant. However, for streams with lower dynamics, such as M_3, reducing δ actually improves the PSNR metric, although marginally, since the sequence is not affected by dropped frames.

The QCT parameter controls the frequency of the QoS renegotiation requests. Its impact on the QoS metrics depends strongly on the stream characteristics. When the streams have narrow and strong bandwidth peaks, higher QCT values increase the QoS renegotiation latency, potentially leading to a quality deterioration. This effect can be inferred from Table III (experiments d2 and d4), where an increase in QCT from one to two leads streams M_1 and M_5 to experience a significant increase in the number of dropped frames and, together with M_6, a deterioration in the PSNR figure. In the other streams, that either have bandwidth requirements that stay constant or nearly constant during relatively long periods of time, higher QCT values do not lead to a significant number of dropped frames and, furthermore, help filter spurious changes that could, otherwise, lead to unnecessary QoS renegotiations, as in the case of M_4 in experiments d2 and d4.

TABLE III
DYNAMIC EXPERIMENTAL RESULTS

d1(0.1,1)	M_1	M_2	M_3	M_4	M_5	M_6	*mean*
DrF	9	0	0	0	4	16	4.83
Wb	0.89	0.90	1.0	0.86	0.78	0.98	0.90
PSNR	32.5	34.7	34.3	35.2	32.0	31.9	33.4
QI	0.88	0.91	0.92	0.90	0.87	0.89	0.89
d2(0.05,1)	M_1	M_2	M_3	M_4	M_5	M_6	*mean*
DrF	15	2	0	8	30	17	9.5
Wb	0.4	0.41	0.47	0.39	0.34	0.42	0.40
PSNR	32.6	34.7	34.6	35.2	32.9	32.9	33.81
QI	0.88	0.91	0.92	0.90	0.87	0.89	0.89
d3(0.1,2).	M_1	M_2	M_3	M_4	M_5	M_6	*mean*
DrF	6	0	0	0	5	15	4.3
Wb	0.88	0.90	1.0	0.87	0.78	0.98	0.90
PSNR	32.5	34.7	34.3	35.2	32.0	31.8	33.41
QI	0.88	0.91	0.92	0.90	0.87	0.89	0.91
d4(0.05,2)	M_1	M_2	M_3	M_4	M_5	M_6	*mean*
DrF	24	2	0	4	36	17	13.83
Wb	0.41	0.41	0.48	0.40	0.34	0.43	0.41
PSNR	32.5	34.7	34.6	35.2	32.0	31.7	33.45
QI	0.88	0.91	0.92	0.90	0.87	0.89	0.89
d5 (0.05,1)	M_1	M_2	M_3	M_4	M_5	M_6	*mean*
DrF	24	0	2	11	27	24	14.6
Wb	0.47	0.30	0.35	0.25	0.45	0.55	0.39
PSNR	33.0	33.6	33.4	33.7	32.7	32.6	33.16
QI	0.89	0.90	0.89	0.87	0.87	0.90	0.88

One aspect that should be highlighted is the low sensitivity of the system to the particular values of δ and QCT. In fact, the PSNR and QI metrics do not change significantly with any of these parameters, thus facilitating system setup.

Comparing Tables III and IV clearly shows that the dynamic approach leads to significant improvements in all key aspects.

TABLE IV
STATIC EXPERIMENTAL RESULTS

s1.	M_1	M_2	M_3	M_4	M_5	M_6	*mean*
DrF	7	0	0	0	2	10	3.16
Wb	1.0	1.38	1.1	1.66	1.09	0.55	1.16
PSNR	29.8	32.7	31.16	33.48	29.12	29.25	30.91
QI	0.82	0.88	0.89	0.88	0.8	0.85	0.83
s2	M_1	M_2	M_3	M_4	M_5	M_6	*mean*
DrF	47	0	0	0	16	87	50
Wb	0.78	1.2	0.91	1.48	0.88	0.3	0.93
PSNR	32.66	34.37	33.87	34.78	32.38	32.0	33.34
QI	0.88	0.90	0.91	0.90	0.87	0.89	0.89
s3.	M_1	M_2	M_3	M_4	M_5	M_6	*mean*
DrF	1871	2	2	2	60	60	332.8
Wb	1.54	0.93	0.64	1.23	0.62	0.62	0.93
PSNR	25.24	34.6	34.12	34.98	32.36	32.05	32.22
QI	0.68	0.90	0.91	0.90	0.87	0.89	0.85
s4.	M_1	M_2	M_3	M_4	M_5	M_6	*mean*
DrF	11	0	0	0	1	41	8.83
Wb	1.04	1.37	1.06	1.68	1.18	0.64	1.12
PSNR	30.55	31.23	30.66	31.88	30.82	30.00	30.85
QI	0.82	0.83	0.85	0.84	0.85	0.83	0.85

TABLE V
QoS RESULTS

	s1	**s2**	**s3**	**s4**	
QoS'	0.41	0.48	0.46	0.40	
	d1	**d2**	**d3**	**d4**	**d5**
QoS'	0.47	0.64	0.47	0.63	0.67

The number of dropped frames is strongly reduced, mainly in the streams with higher dynamics (e.g., M_1 and M_5). The quality metrics (PSNR and QI) are also consistently similar or better. It should be remarked that these results are achieved with better bandwidth utilization. The exception is for s2, which, with a constant $q = 55$, attains similar quality levels, with a low number of dropped frames. In fact, it is possible, in some cases, to find the best static q for each stream. However, this procedure has to be done offline (requiring *a priori* knowledge of the stream), thus not being suitable for MES applications.

Table V presents the QoS' values for each experiment. The first conclusion that can be withdrawn is that, for properly selected δ parameters, the QoS attained with the dynamic approach can be significantly higher than that with the static approach, e.g., d2 and d4 versus s2 and s4. Considering the meaning of this metric, one can conclude that higher quality levels can be attained both by allocating more bandwidth to the streams that can make better use of it, as well as by reducing the wasted bandwidth. The impact of the wasted bandwidth in this metric can also be observed in the significant difference, around 35%, between experiments d1 and d2, and d3 and d4.

Experiment d5 aims at illustrating the system behavior when the assigned priorities are not uniform. The Pr values used in d5 imply a bandwidth distribution where streams M_1, M_5, and M_6 obtain more resources in detriment of streams M_2, M_3, and M_4, as can be seen in Fig. 5. This matches the requirements of many applications in which some streams have a higher impact on the global system performance and thus should be favored. Fig. 6 shows the bandwidth used by stream M_1 in experiments d1, d2, and d5. It can be observed that, in experiment d5, the scheduler assigns more bandwidth to stream M_1 than in experiments d1 and d2. This observation is particularly clear when comparing experiments d2 and d5, which have an equivalent parameterization except for the priority. Observing Table III,

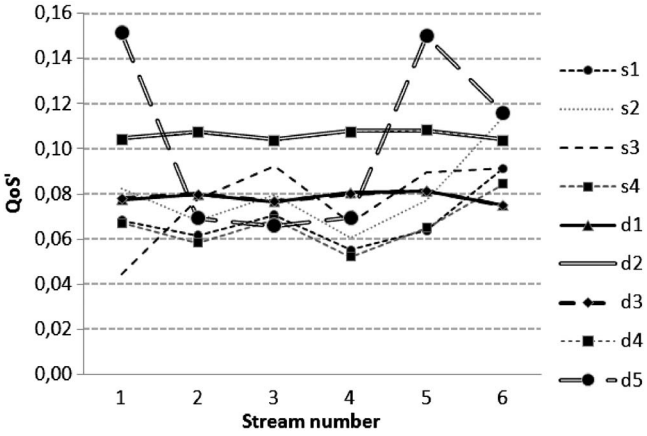

Fig. 5. Contribution of each stream to QoS'.

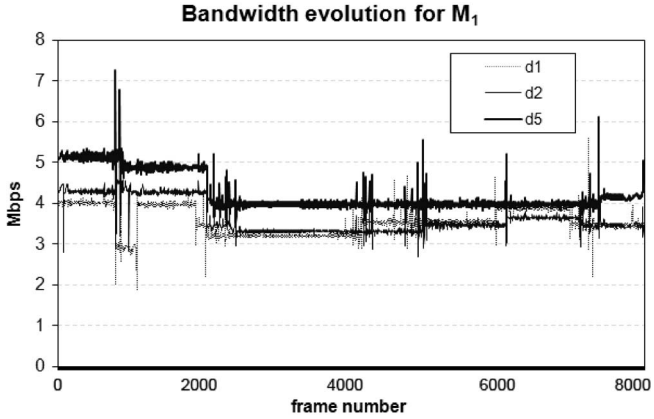

Fig. 6. Stream 1 bandwidth evolution.

it is possible to conclude that the higher priority streams have a gain between 0.5 and 1 dB, at the expense of a decrease between 1 and 1.2 dB in the lower priority ones. Thus, the priority mechanism proves its effectiveness in differentiating the streams, providing more resources to the ones that have higher impact in the global system performance.

Finally, note that the use of prerecorded video sequences instead of cameras was transparent to the operation of the system and that, as expected, no performance bottleneck was found despite the frequent QoS adaptations (adaptations of q) and occasional channel bandwidth renegotiations [changes in (C, T)].

V. CONCLUSION

Using multimedia streams in real-time applications requires appropriate support from the underlying network. A common technique has been to allocate CBR channels to different streams, favoring temporal isolation. However, multimedia streams are intrinsically of the VBR type, so there is either a degradation on the quality, if the channels are designed for the average requirements, or a significant bandwidth waste, if the channels are designed to fit worst case requirements. This paper has proposed a multidimensional dynamic QoS adaptation mechanism that allows dynamically changing the channel bandwidth according to the effective stream needs and overall available bandwidth. This adaptation mechanism is extensively assessed, with its performance being compared against a corresponding situation with static CBR channels, using a set of

stored video sequences from industrial environments. A new QoS metric that considers both the image quality, stream priorities, and the capacity of the system to reduce wasted bandwidth is used to assess the performance. The results obtained show a consistent superiority of the dynamic adaptation mechanism, particularly when there are streams of different priorities. Moreover, such adaptation is carried out with reserved channels, thus maintaining the temporal isolation feature among the streams and other real-time traffic, thus being suitable for integration in complex MESs, integrating real-time sources of diverse natures, e.g., including closed-loop feedback control.

REFERENCES

[1] F. G. Molinero, "Real-time requirements of media control applications," in *Proc. 19th Euromicro Conf. Real-Time Syst.*, Pisa, Italy, Jun. 2007.

[2] C.-S. Cho, B.-M. Chung, and M.-J. Park, "Development of real-time vision-based fabric inspection system," *IEEE Trans. Ind. Electron.*, vol. 52, no. 4, pp. 1073–1079, Aug. 2005.

[3] A. Kumar, "Computer-vision-based fabric defect detection: A survey," *IEEE Trans. Ind. Electron.*, vol. 55, no. 1, pp. 348–363, Jan. 2008.

[4] P. Vadakkepat, P. Lim, L. C. De Silva, L. Jing, and L. L. Ling, "Multimodal approach to human-face detection and tracking," *IEEE Trans. Ind. Electron.*, vol. 55, no. 3, pp. 1385–1393, Mar. 2008.

[5] C.-L. Hwang and C.-Y. Shih, "A distributed active-vision network-space approach for the navigation of a car-like wheeled robot," *IEEE Trans. Ind. Electron.*, vol. 56, no. 3, pp. 846–855, Mar. 2009.

[6] W.-F. Xie, Z. Li, X.-W. Tu, and C. Perron, "Servoing with laser pointer in robotic manufacturing systems," *IEEE Trans. Ind. Electron.*, vol. 56, no. 2, pp. 520–529, Feb. 2009.

[7] Y. Motai and A. Kosaka, "Hand–eye calibration applied to viewpoint selection for robotic vision," *IEEE Trans. Ind. Electron.*, vol. 55, no. 10, pp. 3731–3741, Oct. 2008.

[8] B. Rinner and W. Wolf, "An introduction to distributed smart cameras," *Proc. IEEE*, vol. 90, no. 10, pp. 1565–1575, Oct. 2008.

[9] L. Almeida, P. Pedreiras, and J. A. Fonseca, "The FTT-CAN protocol: Why and how," *IEEE Trans. Ind. Electron.*, vol. 49, no. 6, pp. 1189–1201, Dec. 2002.

[10] P. Pedreiras, P. Gai, L. Almeida, and G. Buttazzo, "FTT-Ethernet: A flexible real-time communication protocol that supports dynamic QoS management on Ethernet-based systems," *IEEE Trans. Ind. Informat.*, vol. 1, no. 3, pp. 162–172, Aug. 2005.

[11] J. Decotignie, "The many faces of industrial Ethernet (past and present)," *IEEE Ind. Electron. Mag.*, vol. 3, no. 1, pp. 8–19, Mar. 2009.

[12] J. Jasperneite, J. Imtiaz, M. Schumacher, and K. Weber, "A proposal for a generic real-time Ethernet system," *IEEE Trans. Ind. Informat.*, vol. 5, no. 2, pp. 75–85, May 2009.

[13] J.-P. Thomesse, "Fieldbus technology in industrial automation," *Proc. IEEE*, vol. 93, no. 6, pp. 1073–1101, Jun. 2005.

[14] J. Silvestre, L. Almeida, R. Marau, and P. Pedreiras, "Dynamic QoS management for multimedia real-time transmission in industrial environments," in *Proc. 12th IEEE Int. Conf. Emerging Technol. Factory Autom.*, Sep. 2007, pp. 1473–1480.

[15] *Information Technology—Digital Compression and Coding of Continuous-Tone Still Images: Requirements and Guidelines*, ISO/IEC 10918-1, Feb. 1994.

[16] M. W. Marcellin, M. J. Gormish, A. Bilgin, and M. P. Boliek, "An overview of JPEG-2000," in *Proc. DCC*, Mar. 2000, pp. 523–541.

[17] *Generic Coding of Moving Pictures and Associated Audio Information—Part 2, Video*, ISO/IEC DIS 13818-2, May 1994.

[18] *Video Coding for Low Bit Rate Communications*, ITU-T Recommendation H.263, Apr. 1995.

[19] *Coding of Audio-Visual Objects—Part 2: Visual*, ISO/IEC 14496-2 (MPEG-4 Visual Version 1), Apr. 1999.

[20] *Coding of Audio-Visual Objects—Part 10: Advanced Video Coding (AVC) ISO/IEC 14496-10*, ITU-T Recommendation H.264 AVC for generic audio visual services, 2003.

[21] *White Paper: Digital Video Compression: Review of the Methodologies and Standards to Use for Video Transmission and Storage*, Axis Commun., Lund, Sweden, Jun. 2004.

[22] B. Bouras and A. Gkamas, "Multimedia transmission with adaptative QoS based on real-time protocols," *Int. J. Commun. Syst.*, no. 16, pp. 225–248, 2003.

[23] B. Vandalore, W. Feng, R. Jain, and S. Fahmy, "A survey of application layer techniques for adaptive streaming of multimedia," *Real-Time Imaging*, vol. 7, no. 3, pp. 221–235, Jun. 2001.

[24] V. Veeraraghavan and S. Weber, "Fundamental tradeoffs in distributed algorithms for rate adaptive multimedia streams," *Comput. Netw.*, vol. 52, no. 6, pp. 1238–1251, Apr. 2008.

[25] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications," RFC 1889, Audio-Video Transport Working Group, Sep. 1987.

[26] H. Schulzrinne, A. Rao, and R. Lanphier, "Real Time Streaming Protocol (RTSP)," RFC 2326, Audio-Video Transport Working Group, Apr. 1998.

[27] M. Handley, H. Schulzrinne, E. Schooler, and J. Rosenberg, "SIP: Session Initiation Protocol," RFC 2543, Networking Working Group, Mar. 1999.

[28] S. Sen, J. L. Rexford, J. K. Dey, J. F. Kurose, and D. F. Towsley, "Online smoothing of variable-bit-rate streaming video," *IEEE Trans. Multimedia*, vol. 2, no. 1, pp. 37–48, Mar. 2000.

[29] T. Liu and C. Choudary, "Content-adaptive wireless streaming of instructional videos," *ACM Multimedia Tools Appl.*, vol. 28, no. 1, pp. 157–171, Jan. 2006.

[30] A. Mittal, A. Pande, and P. Kumar, "Content-based network resource allocation for real time remote laboratory applications," in *Signal, Image and Video Processing*. New York: Springer-Verlag, 2009.

[31] J. Ribas-Corbera and S. Lei, "Rate control in DCT video coding for low-delay communications," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 9, no. 1, pp. 172–185, Feb. 1999.

[32] J.-C. Tsai, "Rate control for low-delay video coding using a dynamic rate table," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 15, no. 1, pp. 133–137, Jan. 2005.

[33] R. Razavi, M. Fleury, and M. Ghanbari, "Low-delay video control in a personal area network for augmented reality," *IET Image Process.*, vol. 2, no. 3, pp. 150–162, Jun. 2008.

[34] *Video Codec Test Model*, TMN8, ITU-T/SG15, 1997.

[35] Z. Zhang, S. Nelakuditi, R. Aggarwal, and R. P. Tsang, "Efficient selective frame discard algorithms for stored video delivery across resource constrained networks," *Real-Time Imaging*, vol. 7, no. 3, pp. 255–273, Jun. 2001.

[36] M. G. Valls, A. Alonso, and J. A. de la Puente, "Dynamic adaptation mechanism in multimedia embedded systems," in *Proc. IEEE Int. Conf. INDIN*, Jun. 2009, pp. 188–193.

[37] R. Marau, P. Leite, M. Velasco, P. Marti, L. Almeida, P. Pedreiras, and J. M. Fuertes, "Performing flexible control on low cost microcontrollers using a minimal real-time kernel," *IEEE Trans. Ind. Informat.*, vol. 4, no. 2, pp. 125–133, May 2008.

[38] R. Marau, L. Almeida, and P. Pedreiras, "Enhancing real-time communication over COTS Ethernet switches," in *Proc. IEEE Int. Workshop Factory Commun. Syst.*, Sep. 2006, pp. 295–302.

[39] W. Ding and B. Liu, "Rate control of MPEG video coding and recording by rate-quantization model," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, no. 2, pp. 12–20, Feb. 1996.

[40] J. Silvestre, V. Sempere, and T. Albero, "Industrial video sequences for network performance evaluation," in *Proc. 5th IEEE Workshop Factory Commun. Syst.*, Sep. 2004, pp. 343–347.

[41] Z. Wang and A. C. Bovik, "Mean squared error: Love it or leave it?" *IEEE Signal Process. Mag.*, vol. 26, no. 1, pp. 98–117, Jan. 2009.

[42] Z. Wang, L. Lu, and A. C. Bovik, "Video quality assessment based structural distortion measurement," *Signal Process.: Image Commun.*, vol. 19, no. 2, pp. 121–132, Feb. 2004.

[43] R. Marau, L. Almeida, P. Pedreiras, M. Gonzalez Harbour, and D. Sangorrin, "Integration of a flexible network in a resource contracting framework," in *Proc. 12th IEEE Conf. Emerging Technol. Factory Autom.*, Sep. 2007, pp. 1481–1488.

Javier Silvestre-Blanes (M'02) received the *Licenciatura* degree in computer engineering and the Ph.D. degree in computer architecture from the Universidad Politécnica de Valencia (UPV), Valencia, Spain, in 1999 and 2003, respectively.

He was a Computer Vision Engineer in a research and development institute in Alcoy, Spain. He is currently an Associate Professor with the Computer Engineering Department, UPV, where he is also a Senior Researcher with the Instituto Tecnológico de Informática. His research interests include real-time networks, multimedia systems, and communications.

Dr. Silvestre-Blanes is a member of the IEEE Industrial Electronics Society.

L. Almeida (S'86–M'89) received the *Licenciatura* degree in electronics and telecommunications engineering and the Ph.D. degree in electrical engineering from the University of Aveiro, Aveiro, Portugal, in 1988 and 1999, respectively.

He was a Design Engineer in a company producing digital telecommunications equipment. He is currently an Associate Professor with the Department of Electrical and Computer Engineering, University of Porto, Porto, Portugal, and a Senior Researcher with the Institute of Electronics and Telematics Engineering of Aveiro and Telecommunications Institute Research Units. His research interests include those related to real-time networks for distributed industrial/embedded systems and control architectures for mobile robots.

Dr. Almeida is a member of the IEEE Computer Society.

R. Marau (M'07) received the *Licenciatura* degree in electronics and telecommunications and the Ph.D. degree in informatics engineering from the University of Aveiro, Aveiro, Portugal, in 2004 and 2009, respectively.

He is currently a Researcher with the Department of Electrical and Computer Engineering, University of Porto, Porto, Portugal. His research interests include real-time scheduling in communication systems, dynamic reconfigurability, and quality-of-service management.

P. Pedreiras (M'03) received the *Licenciatura* degree in electronics and telecommunications engineering and the Ph.D. degree in electrical engineering from the University of Aveiro, Aveiro, Portugal, in 1997 and 2003, respectively.

He is currently an Assistant Professor with the Department of Electronics, Telecommunications and Informatics, University of Aveiro. His main research interests include distributed embedded systems, real-time networks, real-time operating systems, intelligent transportation systems, and mobile robotics.