Quality-Aware Media Scheduling on MPSoC Platforms

Deepak Gangadharan*, Samarjit Chakraborty†, Roger Zimmermann‡
Technical University of Denmark*, TU Munich, Germany†, National University of Singapore‡
E-mail:{dega@imm.dtu.dk, samarjit@tum.de, rogerz@comp.nus.edu.sg}

***Abstract*—Applications that stream multiple video/audio or video+audio clips are being implemented in embedded devices. A Picture-in-Picture (PiP) application is one such application scenario, where two videos are played simultaneously. Although the PiP application is very efficiently handled in televisions and personal computers by providing maximum quality of service to the multiple streams, it is a difficult task in devices with resource constraints. In order to efficiently utilize the resources, it is essential to derive the necessary processor cycles for multiple video streams such that they are displayed with some prespecified quality constraint. Therefore, we propose a network calculus based formal framework to help schedule multiple media streams in the presence of buffer contraints. Further, our framework also presents a schedulability analysis condition to check if the multimedia streams can be scheduled such that a prespecified quality constraint is satisfied with the available service. We present this framework in the context of a PiP application, but it is applicable in general for multiple media streams. The results obtained using the formal framework were further verified using experiments involving system simulation.**

I. Introduction

Simultaneous viewing of multiple video streams has recently become a very common feature in televisions (TVs) and personal computers (PCs). These multiple video streams are either displayed adjacent to each other (as in the display of programs from multiple channels simultaneously on a TV or PC) or as a Picture-in-Picture (PiP) where one video stream is displayed on the full screen while the other is displayed in an inset window. In order to process these multiple streams with maximum quality, adequate number of processor cycles need to be provided. There are several works in literature that analyze multimedia stream processing ([1]- [2]) with the objective of achieving maximum quality, i.e., without frame drops.

Mobile devices are currently providing the PiP application [3] to allow users to watch two videos simultaneously. Recently, a multi-tasking video playback application [4] was introduced in the Android app store. This supports a pop up video playback while the user executes other tasks on the mobile such as web browsing, etc. This also gives rise to a scenario where multiple media streams could be played back simultaneously. However, in contrast to TVs and PCs, these mobile devices have acute resource constraints, which makes it difficult to process multiple video streams with maximum quality when other applications run simultaneously. Although the processors in current mobile devices run at high speeds, it will not be always possible to run the processors at the maximum speed due to power and temperature constraints. In addition to that it is highly likely that the buffer space allocated to the media player applications are sometimes not sufficient to playback the video streams without frame drops. Therefore, it is essential to have a framework that enables the system designer to compute the adequate service requirements such that a target media quality requirement is satisfied.

978-3-9815370-0-0/DATE13/

In this work, we propose a formal framework to design an appropriate scheduler that services the multiple incoming streams in a PiP application such that certain quality constraints are satisfied. This framework will be useful for scheduling multiple multimedia streams in a mobile device, where some frame drops can be tolerated without significant deterioration in the quality of the streams. Our framework also provides a schedulability analysis condition to quickly determine if the available processor service is sufficient to schedule the streams, while adhering to their required quality constraints.

A. Related Work

Processor bandwidth allocation is integral to the desired functioning of the multimedia applications on MPSoC platforms especially due to the intensive computations required for certain multimedia tasks. Therefore, an appropriate processor scheduling algorithm is important for efficiently handling multiple tasks. These algorithms are designed with various design objectives in mind. In [5], scheduling algorithms were discussed to minimize the buffer requirements for multimedia applications. The authors proposed a static priority based scheduling algorithm, which demonstrated the need for smaller buffer size than the other existing scheduling algorithms. Nieh and Lam [6] discussed an integrated scheduling framework to handle both real-time and conventional applications including multimedia with adequate fairness. Hence, in overloaded scenario, the real-time tasks degraded gracefully.

Yuan and Nahrstedt [7] presented a scheduler that accomodates the objective of energy efficiency while scheduling multimedia tasks on mobile devices by integrating dynamic voltage scaling along with soft real-time scheduling policy. There are few works that attempt to reserve some bandwidth for the soft-real time applications like multimedia in the presence of hard real-time tasks and map these tasks on a heterogenous mutliprocessor platform so that the hard real-time tasks are schedulable and the quality of service (QoS) for soft-real time tasks are maximized [8], [9]. However, to the best of our knowledge, the problem of allocating processor bandwidth in the presence of resource constraints such that media quality degradations measured using an *objective quality metric* such as Peak Signal to Noise Ratio (PSNR) are bounded and measurable, has not been sufficiently studied.

Our Contributions: In this work, we derive mathematical bounds for the processor service requirements to process multimedia streams in a quality-aware manner. Once the bounds on processor service requirements are computed, it can be used to derive the parameters of any specific scheduler. Most other works that consider media quality while deriving the service requirements do not ensure a bounded loss in quality, whereas our framework derives service bounds under bounded quality degradation, which is measured in terms of PSNR. The framework is flexible enough to accomodate other objective

quality metrics. In [10], the authors present a mathematical framework to compute the reduced buffer requirements for a MPSoC platform running multimedia decoder under bounded quality constraint and a given processor service allocation for a single multimedia stream. However, in order to compute the mathematical bounds for the required processor service under a bounded quality and buffer constraints, we cannot use the inverse of the mathematical framework presented in [10] for a processor servicing multiple incoming multimedia streams. Therefore, we propose a framework that presents a solution to this problem.

II. Framework Overview

This section presents an overview of our mathematical framework to derive the appropriate processor service requirement (in terms of service bounds) such that the multiple incoming multimedia streams adhere to their individual target quality constraints (in terms of PSNR). Our framework uses the arrival and service curve concepts from Network Calculus [11] to model the data arrival and service given by the resources, respectively. These arrival and service curves efficiently capture the variability in the data arrival and the processing required for the arrived data.

Platform Description: In this work, we analytically derive the scheduler parameters necessary to schedule multiple incoming streams on a resource constrained MPSoC architecture as shown in Fig. 1 with acceptable quality deterioration. The architecture consists of two processing elements (PEs) denoted by PE_1 and PE_2. Each PE services two individual incoming streams $a_1(t)$ and $a_2(t)$, which are cumulative functions that denote the total number of stream objects (such as macroblocks or frames in a video stream) that arrive over the time interval $[0,t]$. Moreover, here we assume that the PEs execute tasks from a video decoder application like MPEG-2. The buffers used by stream $a_1(t)$ have sizes B_1, B_3 and B_{a1} (in number of frames) at the input, intermediate and playout stages of the setup, respectively. Similarly, the buffers used by stream $a_2(t)$ have sizes B_2, B_4 and B_{a2} (in number of frames) at the input, intermediate and playout stages of the setup, respectively. $y_1(t)$ and $y_2(t)$ are the processed stream outputs from PE_1 corresponding to inputs $a_1(t)$ and $a_2(t)$, respectively. Similarly, $z_1(t)$ and $z_2(t)$ are the processed stream outputs from PE_2 corresponding to inputs $y_1(t)$ and $y_2(t)$, respectively. In our setup, $y_1(t)$, $y_2(t)$, $z_1(t)$ and $z_2(t)$ are also cumulative functions. The playout consumption functions for the two streams are denoted as $C_1(t)$ and $C_2(t)$, respectively, which are also cumulative functions.

Definition 1: **(Frame Interval)**. For a given video clip, a frame interval F is defined as a window of any F consecutive frames.

Definition 2: **(Arrival Curve)**. For a video clip, let $a(t)$ denote the number of frames that arrive in time interval $[0,t)$. Then, the video clip is said to be bounded by the arrival curve $\alpha = [\alpha^u, \alpha^l]$ iff for all arrival patterns $a(t)$:

$$\alpha^l(\Delta) \leq a(t+\Delta) - a(t) \leq \alpha^u(\Delta) \tag{1}$$

for all $\Delta \geq 0$. In other words, $\alpha^u(\Delta)$ and $\alpha^l(\Delta)$ give the maximum and minimum number of frames that can arrive over any interval of length Δ across the length of the video clip.

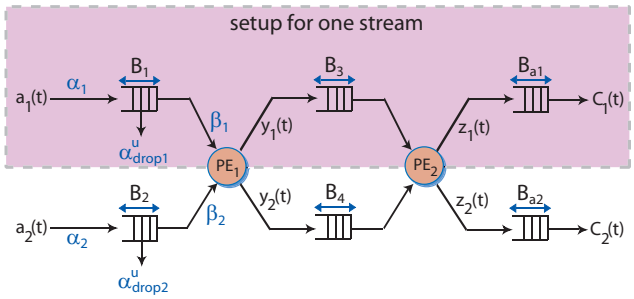


Fig. 1. MPSoC platform setup for a PiP-like application with frame drops showing two streams with separate buffers, but sharing processing resources.

Definition 3: **(Service Curve)**. Let $c(t)$ denote the number of frames processed by a task mapped onto a processor in time interval $[0,t)$. Then, the service curve $\beta = [\beta^u, \beta^l]$ is a service curve of the processor iff for all service patterns $c(t)$:

$$\beta^l(\Delta) \leq c(t+\Delta) - c(t) \leq \beta^u(\Delta) \tag{2}$$

for all $\Delta \geq 0$. In other words, $\beta^u(\Delta)$ and $\beta^l(\Delta)$ denote the upper and lower bounds on the number of frames processed over any interval of time Δ across the length of the clip.

In our setup, $\alpha_1 = [\alpha_1^u, \alpha_1^l]$ and $\alpha_2 = [\alpha_2^u, \alpha_2^l]$ are the arrival curves for the two streams at the input stage of the architecture as shown in Fig. 1, while $\beta_1 = [\beta_1^u, \beta_1^l]$ and $\beta_2 = [\beta_2^u, \beta_2^l]$ are the service curves offered to the two streams by PE_1.

Problem Definition: *Given the arrival curves α_1 and α_2, corresponding to the two video streams ($a_1(t)$ and $a_2(t)$, respectively) in a PiP-like application that are required to be decoded on a resource-constrained MPSoC platform, the sizes of input buffers (B_1 and B_2), the sizes of intermediate buffers (B_3 and B_4), the sizes of playout buffers (B_{a1} and B_{a2}) and the playout consumption functions ($C_1(t)$ and $C_2(t)$), we analytically derive the mathematical bounds on scheduler service of PE_1, such that the two video streams individually satisfy their respective target quality constraints, Q_1^{target} and Q_2^{target} (in PSNR). Consequently, the scheduler parameters can be derived from the service bounds.*

The first stage encounters frame drops as shown in Fig. 1, where the number of frame drops in any time interval Δ is bounded by $\alpha^u_{drop}(\Delta)$ called *Drop Bound*. The drop bounds for the two incoming streams are denoted by $\alpha^u_{drop1}(\Delta)$ and $\alpha^u_{drop2}(\Delta)$. In order to analyze the MPSoC platform shown in Fig. 1, where there are frame drops, we cannot directly follow the analysis method presented in earlier works (e.g., [1]), which ensure that no buffer overflows and playout buffer does not underflow. In our analysis with frame drops (or buffer

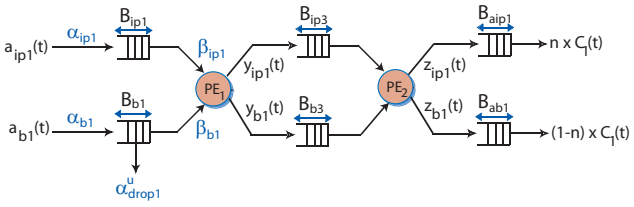


Fig. 2. System model for the shaded portion representing data path for stream $a_1(t)$ in Fig. 1.

overflows), we use the system model shown in Fig. 2 for the shaded portion in Fig. 1. The analysis of the other half is similar to that of the shaded portion.

The incoming video stream is divided into two parts - a significant part denoted by $a_{ip1}(t)$ (no frame drops) and a less significant part denoted by $a_{b1}(t)$ (with frame drops) in Fig. 2. These two parts combine to form the original stream $a_1(t)$, i.e., $a_1(t) = a_{ip1}(t) + a_{b1}(t), \forall t \geq 0$. The partitioned arrival curves corresponding to the two parts are shown as $\alpha_{ip1} = [\alpha^u_{ip1}, \alpha^l_{ip1}]$ and $\alpha_{b1} = [\alpha^u_{b1}, \alpha^l_{b1}]$, respectively, while the partitioned service curves are $\beta_{ip1} = [\beta^u_{ip1}, \beta^l_{ip1}]$ and $\beta_{b1} = [\beta^u_{b1}, \beta^l_{b1}]$, respectively. As shown in Fig. 2, the buffer sizes for the two parts at the three stages of the architecture are $[B_{ip1}, B_{ip3}, B_{aip1}]$ and $[B_{b1}, B_{b3}, B_{ab1}]$, respectively, such that $B_1 = B_{ip1} + B_{b1}$, $B_3 = B_{ip3} + B_{b3}$ and $B_{a1} = B_{aip1} + B_{ab1}$. The partitioned outputs from PE_1 are $y_{ip1}(t)$ and $y_{b1}(t)$, while the partitioned outputs from PE_2 are $z_{ip1}(t)$ and $z_{b1}(t)$ as shown in Fig. 2.

Partitioned Consumption: If the display rate or consumption function at the playout stage for stream $a_1(t)$ is $C_1(t)$, then the consumption functions for the partitioned streams are $n \times C_1(t)$ and $(1-n) \times C_1(t)$ (as shown in Fig. 2), where n is the fraction corresponding to significant part. For example, for a frame sequence of IBBPBBPBBPBBI..., $n = \frac{1}{3}$. If the display rate is 30 frames/second, then the partitioned consumption functions are 10 frames/second and 20 frames/second for I/P frames (significant part) and B frames (less significant part), respectively.

III. Computing Quality-Driven Service Curves

The main objective is to find the bounds for the original service curves $\beta_1 = [\beta^u_1, \beta^l_1]$, which is quality driven. In order to compute these bounds, we first need to find the bounds on the partitioned service curves $\beta_{ip1} = [\beta^u_{ip1}, \beta^l_{ip1}]$ (significant part) and $\beta_{b1} = [\beta^u_{b1}, \beta^l_{b1}]$ (less significant part). For two functions f and g belonging to the set of monotonic functions:
The (min,+) convolution $\otimes$ and deconvolution $\oslash$ operators are defined as $(f \otimes g)(t) = \inf\{f(s) + g(t-s) \mid 0 \leq s \leq t\}$ and $(f \oslash g)(t) = \sup\{f(t+u) - g(u) \mid u \geq 0\}$, respectively. Similarly, the (max,+) convolution $\overline{\otimes}$ and deconvolution $\overline{\oslash}$ operators are defined as $(f \overline{\otimes} g)(t) = \sup\{f(s) + g(t-s) \mid 0 \leq s \leq t\}$ and $(f \overline{\oslash} g)(t) = \inf\{f(t+u) - g(u) \mid u \geq 0\}$, respectively.

The computation of service curve for the significant part follows the method presented in [1]. The β^l_{ip1} bound is therefore obtained by ensuring that buffer B_{ip1} does not overflow, i.e.,

$$\begin{aligned} & a_{ip1}(t) - B_{ip1} \leq y_{ip1}(t), \forall t \geq 0 \\ \Leftrightarrow\ & \beta^l_{ip1}(t) \otimes a_{ip1}(t) \geq a_{ip1}(t) - B_{ip1}, \forall t \geq 0 \\ \Leftrightarrow\ & \beta^l_{ip1}(t) \geq (a_{ip1}(t) - B_{ip1}) \oslash a_{ip1}(t), \forall t \geq 0. \end{aligned} \quad (3)$$

We can compute the β^u_{ip1} bound by ensuring that buffer B_{ip3} does not overflow, i.e.,

$$y_{ip1}(t) \leq z_{ip1}(t) + B_{ip3}, \forall t \geq 0 \quad (4)$$

In order to ensure that buffer B_{aip1} does not underflow, we have

$$z_{ip1}(t) \geq n \times C_1(t), \forall t \geq 0 \quad (5)$$

From Eq. 4 and Eq. 5, in order to strictly ensure that B_{ip3} does not overflow, we can deduce that

$$\begin{aligned} & y_{ip1}(t) \leq n \times C_1(t) + B_{ip3}, \forall t \geq 0 \\ \Leftrightarrow\ & \beta^u_{ip1}(t) \otimes a_{ip1}(t) \leq n \times C_1(t) + B_{ip3}, \forall t \geq 0 \\ \Leftrightarrow\ & \beta^u_{ip1}(t) \leq (n \times C_1(t) + B_{ip3}) \oslash a_{ip1}(t), \forall t \geq 0. \end{aligned} \quad (6)$$

Before we compute $\beta_{b1} = [\beta^u_{b1}, \beta^l_{b1}]$, let us define some quantities that will be used hereafter in this chapter.

Definition 4: **Worst-case quality metric (Q^u)**. For any frame interval F, the worst-case quality metric $Q^u(f,F)$, for all $0 \leq f \leq F_b$, is the worst-case quality of the video if f frames are dropped in any window of F consecutive frames. Here, F_b is the total number of less significant frames that can be dropped and $F_b < F$.

All dropped frames are replaced by immediately preceding and successfully processed frames called *concealment frames*. The amount of quality loss depends on the mean square error (MSE) between the dropped and concealment frames. The MSE and the corresponding PSNR quality are computed as explained in Section V-B in [12].

Definition 5: **Frame interval based time bound ($\delta^u(F)$)**. Given the original arrival curve before partitioning α_1, the upper bound on time required for arrival of F frames is given by

$$\delta^u(F) = \min\{\Delta \geq 0 \mid \alpha^l_1(\Delta) \geq F\}$$

Lemma 3.1: Given the target quality constraint Q^{target}_1, the worst-case quality surface $Q^u(f,F)$ and the frame interval based time bound $\delta^u(F)$, the upper bound on number of frames that can be dropped in any time interval Δ is given by

$$f^u(\Delta) = f_{max}(F)$$

where $f_{max}(F)$ is the maximum number of frames that can be dropped in a frame interval F such that $Q^u(f_{max}(F),F) \geq Q^{target}_1$ and $\delta^u(F) \leq \Delta < \delta^u(F+1)$.

Proof: This lemma can be proved by considering two instances of time intervals.
Case I: Let us first consider the straightforward case with time intervals given by $\Delta = \delta^u(F)$. These are the lowest time interval values where $\alpha^l_{b1} \geq F$. If $f_{max}(F)$ is the maximum possible number of frames that can be dropped in a frame interval F such that the quality constraint is satisfied, i.e., $Q^u(f_{max}(F),F) \geq Q^{target}_1$, then for $\Delta = \delta^u(F)$, we have $f^u(\Delta) = f_{max}(F)$.
Case II: Now let us consider the time intervals given by $\delta^u(F) < \Delta < \delta^u(F+1)$. These are the time intervals when $\alpha^l_1(\Delta) > F$ and $\alpha^l_1(\Delta) < F+1$. In these time intervals, the maximum number of frames that can be dropped should be at most $f_{max}(F)$ so that the quality constraint is satisfied. If the number of frame drops exceeds $f_{max}(F)$, then the quality constraint is violated because $\alpha^l_1(\Delta) < F+1$ and therefore any frame drop $f_d > f_{max}(F)$ will result in $Q^u(f_d,F) < Q^{target}_1$. Then, for $\delta^u(F) < \Delta < \delta^u(F+1)$, we have $f^u(\Delta) = f_{max}(F)$. Hence, it is proved that $f^u(\Delta) = f_{max}(F), \forall \Delta \geq 0$. ■

Lemma 3.2: Suppose $\alpha_{b1} = (\alpha^u_{b1}, \alpha^l_{b1})$ are the arrival curves of the less significant stream as shown in Fig. 2, $\beta_{b1} = (\beta^u_{b1}, \beta^l_{b1})$ are the service curves for the less significant

stream on PE_1, and B is the size of the input buffer. Then, the number of input frames that can be dropped over any interval of length $\Delta \geq 0$ is upper bounded by $\alpha^u_{drop1}(\Delta)$, defined by (Lemma 4.3 in [10])

$$\alpha^u_{drop1} = (\alpha^u_{b1} - \beta^l_{v1}) \overline{\otimes} 0$$

where $\beta^l_{v1} \stackrel{\text{def}}{=} (\alpha^l_{b1} \otimes \beta^l_{b1} + B_{b1})^* \otimes \alpha^u_{b1}$.

Lemma 3.3: Define α_{b1}, β_{b1} and α^u_{drop1} as in Lemma 3.2. Also define f^u as in Lemma 3.1. Then, for any given time interval $\Delta \geq 0$, in order to satisfy the quality constraint, the lower service curve (β^l_{b1}) is given by

$$\beta^l_{b1}(\Delta) \geq ((((((\alpha^u_{b1} - f^u) \overline{\otimes} 0) \oslash \alpha^u_{b1}) - B_{b1}) \overline{\otimes} 0) \oslash \alpha^l_{b1})(\Delta)$$

Proof: Let us start from the expression for drop bound given in Lemma 3.2. In order to satisfy the quality constraint, the following relation needs to be maintained:

$$\begin{aligned}
& \alpha^u_{drop1}(\Delta) \leq f^u(\Delta) \\
\Leftrightarrow\ & ((\alpha^u_{b1} - \beta^l_{v1}) \overline{\otimes} 0)(\Delta) \leq f^u(\Delta) \\
\Leftrightarrow\ & ((\alpha^u_{b1} - (\alpha^l_{b1} \otimes \beta^l_{b1} + B_{b1})^* \otimes \alpha^u_{b1}) \overline{\otimes} 0)(\Delta) \leq f^u(\Delta) \\
\Leftrightarrow\ & (\alpha^u_{b1} - (\alpha^l_{b1} \otimes \beta^l_{b1} + B_{b1}) \otimes \alpha^u_{b1})(\Delta) \leq f^u(\Delta) \\
& (As\ g \overline{\otimes} 0 \leq h \Rightarrow g \leq h) \\
\Leftrightarrow\ & ((\alpha^u_{b1} - f^u) \overline{\otimes} 0)(\Delta) \leq ((\alpha^l_{b1} \otimes \beta^l_{b1} + B_{b1}) \otimes \alpha^u_{b1})(\Delta) \quad (7)
\end{aligned}$$

The network calculus based transformations can be applied to Eq. 7 to derive the lower bound on β^l_{b1} given by

$$\beta^l_{b1}(\Delta) \geq ((((((\alpha^u_{b1} - f^u) \overline{\otimes} 0) \oslash \alpha^u_{b1}) - B_{b1}) \overline{\otimes} 0) \oslash \alpha^l_{b1})(\Delta)$$

Hence, the lemma is proved. ■

We can compute the β^u_{b1} bound by ensuring that buffer B_{b3} does not overflow, i.e.,

$$y_{b1}(t) \leq z_{b1}(t) + B_{b3} - f^u(t), \forall t \geq 0 \quad (8)$$

In order to ensure that buffer B_{ab1} does not underflow, we have

$$z_{b1}(t) \geq (1-n) \times C_1(t), \forall t \geq 0 \quad (9)$$

From Eq. 8 and Eq. 9, in order to strictly ensure that B_{b3} does not overflow, we can deduce that

$$\begin{aligned}
& y_{b1}(t) \leq (1-n) \times C_1(t) + B_{b3} - f^u(t), \forall t \geq 0 \\
\Leftrightarrow\ & \beta^u_{b1}(t) \otimes a_{b1}(t) \leq (1-n) \times C_1(t) + B_{b3} - f^u(t), \forall t \geq 0 \\
\Leftrightarrow\ & \beta^u_{b1}(t) \leq ((1-n) \times C_1(t) + B_{b3} - f^u(t)) \oslash a_{b1}(t), \forall t \geq 0. \quad (10)
\end{aligned}$$

Lemma 3.4: From Eq. 3, Eq. 6, Eq. 10 and Lemma 3.3, the aggregate service curve $[\beta^u_1, \beta^l_1]$ for the PE allowing drops can be computed as

$$\beta^u_1 \leq \min\{\beta^u_{nd1}, \max\{(\beta^l_{ip1} + \beta^u_{b1}), (\beta^u_{ip1} + \beta^l_{b1})\}\}$$

$$\beta^l_1 \geq \min\{\beta^l_{nd1}, (\beta^u_{ip1} + \beta^l_{b1})\}$$

where β^l_{nd1} is the lower bound on aggregate service curve with no frame drops.

Proof: Let us first consider the lower bound of the aggregate service curve represented by the tuple $[\beta^u_1, \beta^l_1]$ for the PE allowing frame drops (the first PE in our case from Fig. 1). The lower bound β^l_1 should at least service a minimum number of less significant frames such that the number of such frames dropped do not violate the quality constraints. This condition can be satisfied if at least β^l_{b1} less significant frames are serviced. In order to ensure that none of the significant frames are dropped, it is necessary that an additional β^u_{ip1} service is provided. This gives the lower bound on the aggregate service required given by $\{\beta^u_{ip1} + \beta^l_{b1}\}$. However, in order to ensure that β^l_1 does not exceed the lower bound with no frame drops, the appropriate lower bound on aggregate service curve with frame drops is $\beta^l_1 \geq \min\{\beta^l_{nd1}, (\beta^u_{ip1} + \beta^l_{b1})\}$.

Now let us consider the upper bound of the aggregate service curve for the PE allowing frame drops. The upper bound of β^u_1 can be a straightforward sum of the upper bounds of individual service for both significant and less significant frames given by $\beta^u_{ip1} + \beta^u_{b1}$. However, this is a pessimistic estimate. In order to not result in buffer overflow at B_3, the upper bound can be such that $\beta^u_1 \leq \{\beta^l_{ip1} + \beta^u_{b1}\}$ or $\beta^u_1 \leq \{\beta^u_{ip1} + \beta^l_{b1}\}$. But we need to ensure that $\beta^u_1 \geq \beta^l_1$. Therefore, the appropriate β^u_1 required is $\beta^u_1 \leq \max\{(\beta^l_{ip1} + \beta^u_{b1}), (\beta^u_{ip1} + \beta^l_{b1})\}$. However, in order to guarantee that $\beta^u_1 \leq \beta^u_{nd1}$, we take the minimum of the right hand side expressions in the above two inequalities.

Hence, the lemma is proved. ■

However the aggregate service curve can be tuned more accurately if there is an integral relationship between the number of I/P frames and number of B frames. This is demonstrated in the next lemma.

Lemma 3.5: Considering the quantities in Lemma 3.4 and Lemma 3.1, if there is an integral relationship between the number of significant frames and the number of insignificant frames in the stream, i.e., if the ratio $\frac{Number\ of\ less\ significant\ frames}{Number\ of\ significant\ frames} = N$, where N is a positive integer, the aggregate service curve $[\beta^u_1, \beta^l_1]$ can be computed as

$$\beta^u_1 \leq \{\beta^u_{b1} + \frac{\beta^u_{b1} + f^u}{N}\}$$

$$\beta^l_1 \geq \min\{\beta^l_{nd1}, (\beta^l_{b1} + \frac{\beta^l_{b1} + f^u}{N})\}$$

where β^l_{nd1} is as defined in Lemma 3.4.

Proof: As in Lemma 3.4, the lower bound of the aggregate service curve β^l_1 has to process at least β^l_{b1} frames to satisfy quality constraints. As the ratio $\frac{Number\ of\ less\ significant\ frames}{Number\ of\ significant\ frames} = N$, where N is an integer, the maximum number of significant frames that need to be processed in the same time interval are $\frac{\beta^l_{b1} + f^u}{N}$. This is because β^l_{b1} is the minimum number of less significant frames to be processed which does not include those less significant frames that are dropped. So, in order to find the maximum number of significant frames, we need to add the upper bound on number of frames dropped to β^l_{b1} so as to find the total number of less significant frames (Dropped + Processed). The total when divided by N gives the maximum number of significant frames processed, given by $\frac{\beta^l_{b1} + f^u}{N}$. The sum of β^l_{b1} and the previous quantity gives one part of the lower bound. In order to ensure that the lower bound with frame drops does not exceed the lower bound without frame drops (β^l_{nd1}), we derive the lower bound as

$\beta_1^l \geq \min\{\beta_{nd1}^l, (\beta_{b1}^l + \frac{\beta_{b1}^l + f^u}{N})\}$. The same explanation holds for the upper bound when β_{b1}^l is substituted with β_{b1}^u. ■

The service is currently in terms of the number of frames processed in any time interval. This has to be converted into bounds on the number of processor cycles provided in any time interval. Let us now define the tuple $[\sigma^u, \sigma^l]$, which denotes the upper and lower bound on the number of processor cycles provided in a specific time interval Δ. If the maximum number of cycles required to process k frames is denoted by $cmax(k)$, then the bounds on the required processor cycles such that an input stream is processed with target quality constraints is given by

$$\sigma^u(\Delta) \leq cmax(\beta_1^u(\Delta)),$$
$$\sigma^l(\Delta) \geq cmax(\beta_1^l(\Delta)). \quad (11)$$

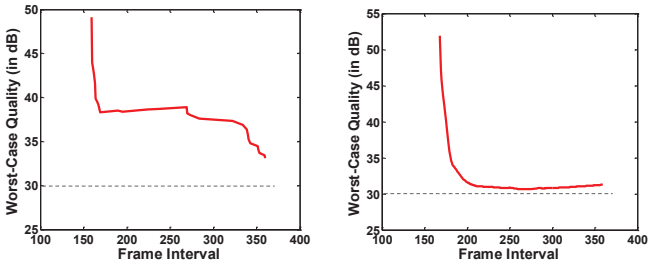
Deriving Scheduler Parameters: Any scheduler that provides service which are bounded by the service bounds $[\sigma^u, \sigma^l]$ that our framework computes is the appropriate scheduler that will process the incoming multimedia stream satisfying a target quality constraint. For example, we illustrate how the scheduler parameters can be derived for a scheduling technique such as Time Division Multiple Access (TDMA). If the total processor cycles available at the PE is c cycles per unit time, then the total processor cycles available is $\sigma^u(\Delta) = \sigma^l(\Delta) = c\Delta$. Let the TDMA scheduler period be P. It allocates $f_1 \times P$ time to stream 1 and $f_2 \times P$ time to stream 2 in a period P such that $f_1 + f_2 \leq 1$. If P is infinitesimally small, then the service curve bounds for stream 1 and stream 2 are $\sigma_1^u(\Delta) = \sigma_1^l(\Delta) = f_1 c\Delta$ and $\sigma_2^u(\Delta) = \sigma_2^l(\Delta) = f_2 c\Delta$, respectively. For a finite P, the service curve bounds become a staircase function. The values of f_1, f_2 and P can be appropriately selected such that Eqn. 11 is satisfied. We could analyze a specific video or a class of video clips.

IV. Experimental Results

In this section, we validate the formal framework presented in the previous section. The two main results presented here are:

1) Verification of the service bounds obtained by checking if the quality constraints are met in a scenario where multiple streams are processed such that each stream is serviced with the processor cycle bounds derived using the formal framework.
2) Reduction in the processor cycle requirements obtained as a result of the trade-off with quality.

In our experiments, we consider frame drops only in front of PE_1. Therefore, we compute the service required on PE_1 for two multimedia streams decoded simultaneously on a MPSoC platform with buffer and processor resource constraints. In particular, we first find the processor cycle bounds in accordance to the formal framework presented in Section III so that target quality constraints for both the multimedia streams are met. Then, we allocate processor cycles to the two streams such that the processor cycle bounds are not violated. The processor cycles required for the multimedia streams on PE_2 (without frame drops) can be computed without partitioning the processed stream at the output of PE_1. The procedure is similar to the computation of processor bounds for I/P frames at PE_1. The cycle requirement for each MPEG-2 and H.264 task is obtained using SimpleScalar sim-profile simulator for a MIPS-like architecture. The results that take advanced microarchitectural features such as pipelining, caches or out-of-order execution into account can be obtained by modifying the SimpleScalar simulator. The MPEG-2 videos (with frame pattern IBBPBBPBBPBBI...) used are *cact*_080 and *susi*_080 (from [13]) and the H.264 videos (with frame pattern IPPPPPPPPP...) used are *nasa* (a clip from NASA) and *bourne_ultimatum* (a movie clip). For MPEG-2 videos (8 Mbps videos), B is the less significant frame and for the H.264 videos (3.25 Mbps videos), the last P frame is the less significant frame. The buffer sizes are fixed at $B_{ip1} = B_{ip2} = 50$, $B_{b1} = B_{b2} = 100$, $B_{ip3} = B_{ip4} = 45$ and $B_{b3} = B_{b4} = 90$ for the MPEG-2 clips and at $B_{ip1} = B_{ip2} = 410$, $B_{b1} = B_{b2} = 40$, $B_{ip3} = B_{ip4} = 415$ and $B_{b3} = B_{b4} = 45$.



(a) (b)

Fig. 3. Simulation results for quality in a multiple stream decoding scenario for (a) *cact*_080 and (b) *susi*_080.

A. Verification of the Computed Service Bounds

The processor cycle requirement curves with frame drops ($[\sigma_1^u, \sigma_1^l]$) obtained in the previous section for both the clips is used in this section to run simulations in a multiple video clip decoding scenario, which is one of the scenarios that can use this framework. PE_1 is assigned a frequency of 500 MHz. The processing cycles are allocated to the video clips in accordance to the cycle requirement bounds $[\sigma_1^u, \sigma_1^l]$ obtained in the previous section. The processor cycles are also allocated in an as late as possible (ALAP) manner such that the video clips are processed at the end of every time interval. This is done in order to ensure that buffer occupancy is the maximum and does not result in quality reduction below the target worst-case quality of 30 *dB*. It is observed from Fig. 3 that the obtained quality for both the videos using the processor cycle bounds does not fall below 30 *dB*, which is the target for the experiment. On the other hand, it is also seen that *susi*_080 achieves a quality much closer to the target worst-case quality of 30 *dB*, while *cact*_080 is a little above 30 *dB*. This is because the variation in video is much higher for *cact*_080 and so the cycle requirements obtained for it are more pessimistic in comparison to those obtained for *susi*_080. The same result was also observed with the H.264 videos for a quality constraint of 32 *dB*.

B. Processor Cycle vs Quality trade-off

We explore this trade-off on PE_1 with buffer and processor bandwidth constraints. In order to compute the processor cycle bounds with no frame drops, we use the method given in [1]. First, we present the result for the aggregate service curve $[\beta_1^u, \beta_1^l]$ with frame drops and compare it with the aggregate

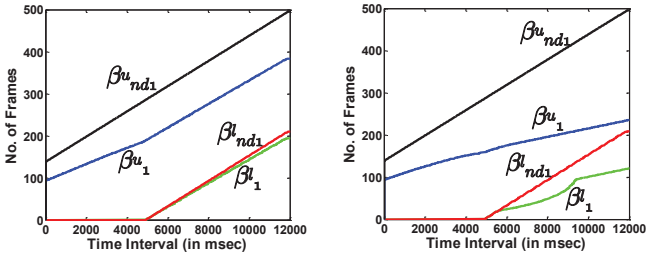


Fig. 4. Aggregate service curves with and without frame drops for the clips (a) *cact*_080 and (b) *susi*_080.

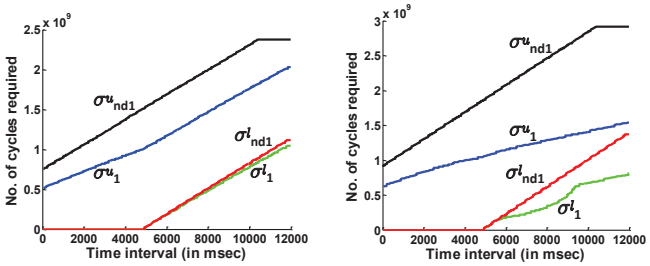


Fig. 5. Processor cycle requirements with and without frame drops for the clips (a) *cact*_080 and (b) *susi*_080.

service curve $[\beta^u_{nd1}, \beta^l_{nd1}]$ without frame drops (for *cact*_080 and *susi*_080) in Fig. 4. In this experiment, frames are dropped such that a worst-case quality of $30dB$ is not violated. It is observed from the graph that the lower service curves with (β^l_1) or without (β^l_{nd1}) frame drops start processing only after an initial latency during which the buffer is not full and there is no frame drop and therefore no loss in quality. After that initial latency, both the curves increase at different rates because with a tolerable loss constraint, β^l_1 need not process all the frames and some frames can be dropped. The reduction in the upper service curve is also observed for the frame drop case (β^u_1) because even though β^u_{nd1} will not cause buffer overflows in the intermediate stage, due to frame drops in the first stage, the upper aggregate service curve β^u_1 decreases as shown in Fig. 4. The observations listed above are seen for all the video clips used to conduct experiments. However, it is evident from Fig. 4(b) that the reduction in service is more for *susi*_080 in comparison to *cact*_080 because the adjacent frames in *susi*_080 are more similar in comparison to the adjacent frames in *cact*_080, which allows more frames to be dropped for *susi*_080 with the same target worst-case quality constraint.

We also plot the curves for the processor cycle requirements for each video clip with a worst-case quality constraint of 30 dB (shown as the tuple $[\sigma^u_1, \sigma^l_1]$) and compare it with the processor cycle requirements without any quality loss (no frame drops) (shown as the tuple $[\sigma^u_{nd1}, \sigma^l_{nd1}]$) in Fig. 5 and Fig. 6. The characteristics of the aggregate service curves is reflected in these curves also as the processor cycle requirements increase with the increase in service requirements. The processor cycle savings for MPEG-2 videos at 30 dB were 40.8% (for *susi*_080) and 6.6% (for *cact*_080), while the same for H.264 videos at 32 dB were 74.63% (for *nasa*) and 34.25% (for *bourne_ultimatum*).

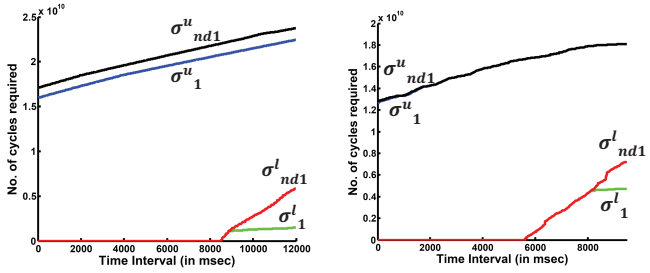


Fig. 6. Processor cycle requirements with ($\geq$ 32 dB) and without frame drops for the clips (a) *nasa* and (b) *bourne_ultimatum*.

V. Concluding Remarks

In this paper, we present a mathematical framework that derives the service bounds required to process a video clip such that a target quality constraint is adhered to. In our framework, we determine service bounds under bounded quality degradation measured using the objective quality metric PSNR. The framework was verified in a multiple video processing setup like a PiP application where the analytically obtained processor service bounds processed the videos, while meeting the target quality constraints.

References

[1] A. Maxiaguine, S. Kunzli, L. Thiele, and S. Chakraborty, "Evaluating schedulers for multimedia processing on buffer-constrained soc platforms," *IEEE Design & Test of Computers*, vol. 21, no. 5, pp. 368–377, 2004.

[2] B. Raman, S. Chakraborty, O. W. Tsang, and S. Dutta, "Reducing data-memory footprint of multimedia applications by delay redistribution," in *44th Design Automation Conference (DAC)*, 2007, pp. 738–743.

[3] "Samsung brings pip to mobile phones," http://gizmodo.com/252919/samsung-brings-pip-to-mobile-phones.

[4] "Stick it video player," https://play.google.com/store/apps/details?id=com.myboyfriendisageek.stickit&hl=en.

[5] S. Rampal, D. P. Agrawal, and D. S. Reeves, "Processor scheduling algorithms for minimizing bu er requirements in multimedia applications," 1994.

[6] J. Nieh and M. S. Lam, "Integrated processor scheduling for multimedia," in *Network and Operating Systems Support for Digital Audio and Video*, 1995, pp. 202–205.

[7] W. Yuan and K. Nahrstedt, "Energy-efficient cpu scheduling for multimedia applications," *ACM Transactions on Computer Systems (TOCS)*, vol. 24, no. 3, pp. 292–331, 2006.

[8] P. Saraswat, P. Pop, and J. Madsen, "Task mapping and bandwidth reservation for mixed hard/soft fault-tolerant embedded systems," in *16th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2010, pp. 89–98.

[9] L. Abeni and G. Buttazzo, "Integrating multimedia applications in hard real-time systems," in *19th IEEE Real-Time Systems Symposium (RTSS)*, 1998, pp. 4–13.

[10] D. Gangadharan, L. Phan, S. Chakraborty, R. Zimmermann, and I. Lee, "Video quality driven buffer sizing via frame drops," in *17th IEEE International Conference on Embedded and Real-Time Computing Systems (RTCSA)*, 2011, pp. 319–328.

[11] J. Le Boudec and P. Thiran, *Network calculus: a theory of deterministic queuing systems for the internet*. springer-Verlag, 2001.

[12] D. Gangadharan, S. Chakraborty, and R. Zimmermann, "Fast hybrid simulation for accurate decoded video quality assessment on mpsoc platforms with resource constraints," in *16th Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2011, pp. 237–242.

[13] "Mpeg-2 benchmark videos," ftp://ftp.tek.com/tv/test/streams/Element/MPEG-Video/625/.