A Queuing-Theoretic Approach to Task Scheduling and Processor Selection for Video-Decoding Applications

Nicholas H. Mastronarde, *Student Member, IEEE*, and Mihaela van der Schaar, *Senior Member, IEEE*

***Abstract*—We propose a cross-layer design for resource-constrained systems that simultaneously decode multiple video streams on multiple parallel processors, cores, or processing elements. Our proposed design explicitly considers the coder specific application characteristics such as the decoding dependencies, decoding deadlines, and distortion impacts of different video packets (e.g., frames, slices, groups of slices etc.). The key to the cross-layer design is the resource management control plane (RMCP) that coordinates the scheduling and processor selection across the active applications. The RMCP deploys a priority-queuing model that can evaluate the system congestion and predict the total expected video quality for the set of active decoding tasks. Using this model, we develop a robust distortion- and delay-aware scheduling algorithm for video packets. This algorithm aims to maximize the sum of achieved video qualities over all of the decoded video sequences. Additionally, we propose a processor selection scheme intended to minimize the delays experienced by the queued video packets. In this way, the number of missed decoding deadlines is reduced and the overall decoded video quality is increased. We compare queuing-theoretic based scheduling strategies to media agnostic scheduling strategies (i.e., earliest-deadline-first scheduling) that do not jointly consider the decoding deadlines and distortion impacts. Our results illustrate that by directly considering the video application's properties in the design of a video decoding system, significant system performance gains on the order of 4 dB peak-signal-to-noise ratio can be achieved.**

***Index Terms*—Middleware, multiprocessor decoding, multimedia systems, queuing models for systems, real-time system scheduling, video decoding complexity.**

I. Introduction

EMERGING multiprocessor devices are poised to enable the proliferation of real-time video applications that require simultaneously decoding multiple video streams on a single device. Typical applications include multipoint videoconferencing, multicamera surveillance, multiview video streaming, and augmented reality. Due to their real-time requirements and high complexity, however, efficiently designing and implementing these video decoding applications on existing resource-constrained and heterogeneous devices poses numerous challenges. First, we must model the *processing delays*, which depend on the source characteristics, the encoding parameters, the deployed compression algorithm, and the system's processing capabilities. Second, we must model the *queuing delays*, which depend on the processing delays, the competing tasks, the number of processors, the distribution of the processing load across each processor, and the deployed coding structures. Predicting these delays is particularly important for delay-sensitive multimedia applications because the derived video quality depends on processing jobs before their deadlines.

Recognizing the importance of these diverse multimedia applications in future commercial enterprises and the need for systems better suited for multimedia applications, Intel and AMD have introduced processing technologies such as dual- and quad-core processors [1], [2] that enable higher throughput for parallelizable workloads. Also, IBM recently developed the Cell Broadband Engine Architecture [3] that is designed specifically for computationally intensive digital entertainment. IBM's Cell allows developers to program middleware that tailors the processor's operation to the specific application. For example, developers can specify that different processing jobs be distributed across the high-bandwidth synergistic processing elements [3] in a way that minimizes delays in the job queue(s). Moreover, IBM's Cell is a scalable processor architecture; that is, it can be produced with different numbers of processing elements. This scalability makes it suitable for a variety of multimedia-enabled devices.

To exploit the available processing resources and efficiently cope with the diverse and dynamic requirements of decoding applications, resource management becomes of paramount importance. Due to the scalability of video applications that can run at different quality- and complexity-levels, a successful video-centric resource management scheme should be designed with the goal of minimizing the decoded video distortion. We argue that existing approaches to implementing multimedia decoding algorithms on systems have several drawbacks:

1) ***No consideration of time-varying application requirements:*** Current systems often assume worst-case resource utilization for the design and implementation of compression techniques and standards, thereby neglecting the fact that multimedia algorithms require time-varying resources [4], [5], which can differ significantly from the worst-case requirements. Specifically, the workload evolves dynamically over time depending on many factors including the source characteristics (e.g., high-motion, complex textures, object occlusion), the encoded bit rates,

Manuscript received February 9, 2007; revised April 27, 2007. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Anna Hac.

N. H. Mastronarde and M. van der Schaar are with the Department of Electrical Engineering (EE), University of California at Los Angeles (UCLA), Los Angeles, CA 90095–1594 USA (e-mail: nhmastro@ee.ucla.edu; mihaela@ee.ucla.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TMM.2007.906568

the transmission bit rates (if the video is streamed over a network), the error concealment strategies, the sub-pixel motion estimation accuracy, and the coding structures of the various decoding tasks in the system. In order to achieve more efficient resource utilization without compromising the decoded video quality, media-aware resource management schemes must rapidly adapt to the applications' time-varying resource requirements. Timely adaptation becomes even more important when there are multiple decoding tasks competing for the same processing resources.

2) ***Compression agnostic adaptation:*** Existing resource management solutions that do consider dynamic application requirements are, nevertheless, often compression agnostic [6], [7]. In [7], for example, a task's CPU allocation is made to satisfy a statistical cycle demand corresponding to an arbitrary fraction of recent jobs in a sliding window (e.g., 95% of frames). This solution, however, ignores the differential distortion impacts of the various encoded frames that naturally arise from the deployed coding structures of state-of-the-art hybrid codecs. Similarly, existing real-time scheduling algorithms for multimedia systems ignore the application's characteristics. For example, most multimedia systems use a compression agnostic earliest-deadline-first (EDF) scheduling policy that schedules tasks based purely on their relative decoding deadlines [6]. Distortion-aware resource management and scheduling is particularly important when deploying state-of-the-art video coders because decoding bit streams created by these video coders often requires processing large workload bursts that can overload the system. In such a scenario, it is essential that any missed decoding deadlines are mitigated to less important (in terms of distortion impact) video packets (e.g., frames, slices, groups of slices etc.). For instance, in an H.264/AVC based coder, losses should be mitigated to B frames in order to protect the more important I and P frames on which subsequent frames depend. Alternatively, if an I or P frame is not decoded, then the quality of the subsequent frames is severely degraded due to error propagation [8].

3) ***No Models for multiprocessor system performance:*** In the majority of work that considers multiprocessor decoding there are no models that can predict the load across the processors based on the active video decoding tasks in the system [9], [10].

Our contributions to designing a distortion- and delay-aware cross-layer solution for multitask, multiprocessor, video decoding systems are as follows:

1) ***Resource management control plane*** (Section II): We propose a Resource Management Control Plane (RMCP) that coordinates the scheduling and processor selection (see contributions 4 and 5 below) across the various video applications while considering their time-varying characteristics, their complexity-scalability properties and the resulting impact on the multimedia quality. The RMCP's actions are driven by the descriptions of the video applications (see contribution 2) and the decoding system model (see contribution 3).

2) ***General framework for describing codec specific properties*** (Sections II–III): We propose a general framework that can be used for describing codec-specific coding structures, decoding dependencies, deadline distributions, distortion impacts, and workload characteristics. Based on this information, we group each sequence's video packets into several coding-structure-specific priority classes. In our framework, any video sequence encoded by a state-of-the-art video coder can be parameterized to interface with the proposed RMCP. This interoperability allows efficient mapping of new multimedia applications to heterogeneous systems and allows existing and new applications to gracefully scale and fairly compete for available resources.

3) ***Modeling system congestion and video distortion*** (Sections II, IV): Given the parameterized video applications, the RMCP employs a queuing model to evaluate their expected resource requirements with only a few parameters. In particular, we model the system using preemptive M/G/1 priority queues [11]. This queuing model allows the RMCP to predict the system congestion (across all processors) as well as the expected video distortion (across all decoded video sequences). Additionally, the RMCP can use the queuing model to estimate the expected waiting times for each priority class on each processor based on the video applications' characteristics, e.g., the distortion impacts of the various priority classes and the off-line modeled service time distributions for different frame types.

4) ***Distortion- and delay-driven video packet scheduling*** (Section V): We explore several scheduling policies for allocating processor-time to video packets. We demonstrate that conventional EDF and priority-queuing (PQ) scheduling policies are sub-optimal (in terms of achieved video quality) when used for decoding H.264/AVC based bit streams. Subsequently, we propose two new distortion- and delay-driven scheduling policies that jointly consider the decoding distortion and the decoding deadlines when allocating processor time to video packets.

5) ***Delay-driven processor selection*** (Section VI): Based on the expected waiting times computed using the priority-queuing analysis, we assign packets to be decoded on the processor that is expected to have the shortest queuing delay. Importantly, minimizing the packet's queuing delay maximizes the probability that it will be successfully decoded before its decoding deadline expires.

To evaluate the cross-layer RMCP solutions proposed in Sections II–VI, we present and discuss our experimental results in Section VII. Finally, we conclude in Section VIII.

II. Resource Management Control Plane

In this section, we introduce the RMCP that is illustrated in Fig. 1. The RMCP consists of two layers: i) The Application Abstraction Layer (AAL), which stores information about the active video applications (i.e., the codec specific decoding deadlines, distortion impacts, and workload distributions) and ii) The System Abstraction Layer (SAL), which uses the application-specific data from the AAL in order to model the system operation (i.e., the expected queuing and processing delays). Based on the information provided by these two layers, the RMCP is

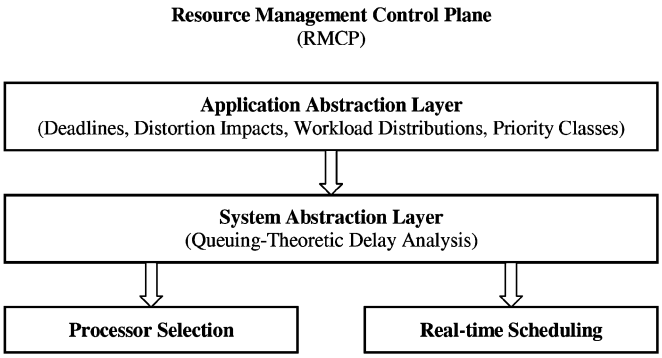

Fig. 1. Block diagram of the Resource Management Control Plane (RMCP).

able to coordinate the real-time scheduling and processor selection by taking into account the applications' time-varying characteristics, complexity-scalability properties, and the resulting impact on the multimedia quality. In the rest of this section, we describe in more detail the functions of the RMCP, AAL, and SAL.

A. Application Abstraction Layer Overview

To enable video sequences encoded by any popular video coders (e.g., MPEG-2, DIVX, MPEG-4, H.264/AVC, wavelet etc.) to interface with the proposed RMCP via the AAL, we desire a general framework for describing codec specific properties such as the delay deadlines, distortion impacts, and the dependencies resulting from the chosen multimedia algorithm. Having a consistent and general interface facilitates efficient mapping of new multimedia applications to heterogeneous systems and allows existing and new applications to gracefully scale and fairly compete for available resources.

We refer to the basic video units in the system as "packets" because video data is often transmitted or stored in a packetized format (e.g., MPEG-21). We let l denote a video packet and we assume that its payload comprises an entire frame, a group of slices within a frame, individual slices, or data partitions, such that the video packets comprising a specific frame are independently decodable. Within the AAL, each video packet l is parameterized as follows:

- ***Decoding deadline:*** τ_l denotes the time by which packet l must be decoded in order for it to contribute to the overall quality of the decoded video.
- ***Distortion reduction:*** Δd_l denotes the decrease in distortion that is observed when the packet is successfully decoded before its decoding deadline. The distortion values may be determined during encoding and transmitted to the decoder as hinting information or metadata [12]. There are many approaches in the literature for determining Δd_l, see e.g., [13] and [14].
- ***Set of Ancestors:*** When decoding a sequence $h(1 \leq h \leq H)$, packets from different frames are processed in decoding order, which is dictated by the dependencies introduced by predictive coding (i.e., motion-estimation and motion-compensation). In general, the dependencies that arise in state-of-the-art video coders for different group of pictures (GOP) structures can be described by directed acyclic graphs (DAGs) [13] with the nodes representing video frames and the edges representing the dependencies between frames. To denote the decoding dependencies among packets, we define A_l as the set of the lth packet's ancestors [15]

$$A_l = \{l' \mid l' \prec l\} \tag{1}$$

where we borrow the notation $l' \prec l$ used in [13] to indicate that l depends on l'.
- ***Indicator of Successful Decoding:*** We use a binary indicator variable w_l that takes value 1 if the lth packet has been decoded and takes value 0 otherwise. If all packets $l' \in A_l$ have been decoded (i.e., $w_{l'} = 1$ for all $l' \in A_l$), then we say that the lth packet is *decodable*. Conversely, as in [13], [15], we assume that if any packet $l' \in A_l$ has not been decoded (i.e., there exists an $l' \in A_l$ such that $w_{l'} = 0$), then l cannot be decoded[1]

Additionally, we define $\mathrm{DAG}(\mathcal{L}^{(h)})$ as the DAG representing the decoding dependencies across an entire sequence. Specifically

$$\mathrm{DAG}(L^{(h)}) = \{A_l \mid l \in \mathcal{L}^{(h)}\} \tag{2}$$

where $\mathcal{L}^{(h)}$ represents the set of packets comprising the hth video sequence $(1 \leq h \leq H)$.

Within the AAL, the packets in the H video sequences are grouped into a total of K priority classes. In general, a video packet's priority increases with the number of packets that are dependent on it. Hence, the video packet prioritization depends on the encoded dependencies encapsulated in $\mathrm{DAG}(\mathcal{L}^{(h)})$, $1 \leq h \leq H$. We denote the kth priority class $(1 \leq k \leq K)$ as q_k and we write $l \in q_k$ if the lth video packet is in priority class q_k. Within each GOP of an encoded video sequence, all of the packets in priority class q_k are assigned a common decoding deadline τ_k (i.e., $\tau_l = \tau_k, \forall\ l \in q_k$ within the same GOP). Successfully decoding every packet in one GOP of priority class q_k yields a distortion reduction Δd_k. Since we assume that the distortion is incrementally additive [13], Δd_k is equivalent to the sum of each packet's distortion reduction, Δd_l, for any $l \in q_k$ within one GOP.

When defining the priority classes, the AAL orders $q_1, \ldots, q_k, \ldots, q_K$ such that $\Delta d_1 \geq \cdots \geq \Delta d_k \geq \cdots \geq \Delta d_K$. Note that, unlike in an embedded bit stream (e.g., a bit stream created by an embedded 3-D wavelet codec), the packets in an H.264/AVC compliant bit stream do not have decreasingly stringent delay requirements as their priority decreases. For this reason, sophisticated hybrid codecs such as H.264/AVC require more complex scheduling algorithms that jointly consider delay deadlines, distortion impacts, and expected processing/service times in order to maximize the decoded quality. This observation motivates the proposed scheduling algorithms in Sections V-C and V-E.

B. System Abstraction Layer Overview

We assume that there are H video decoding tasks sharing the resources of a system comprising M parallel processors. Fig. 2 illustrates the abstraction of the system from the point of view

[1]We do not consider error concealment in our treatment of the problem; however, it can be incorporated into the proposed framework.

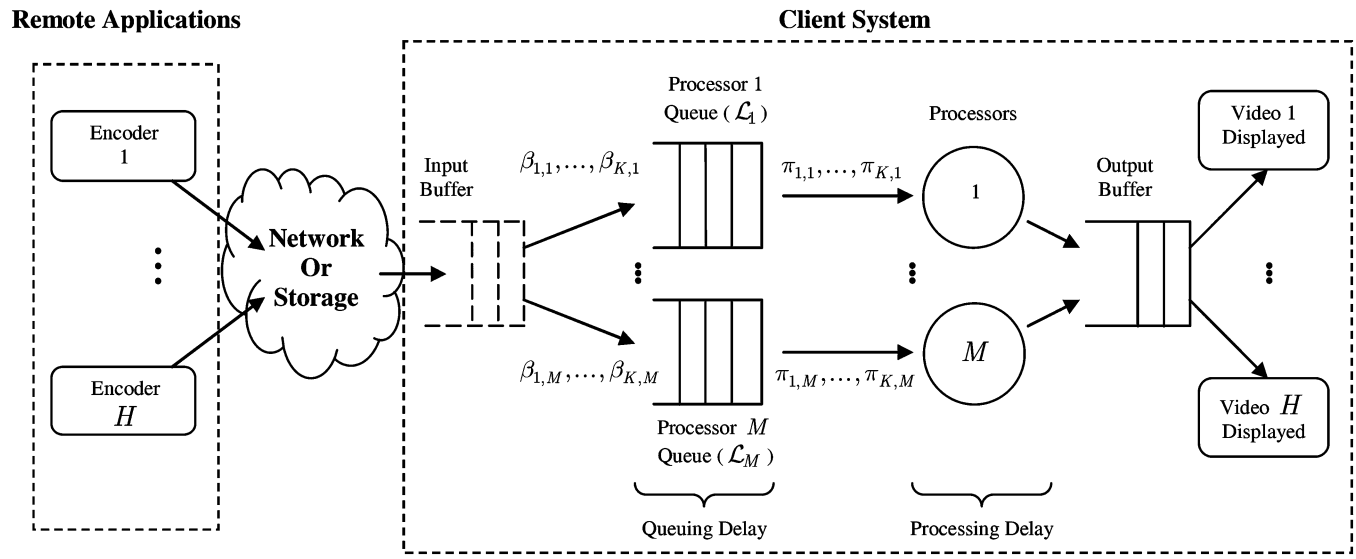

Fig. 2. Internals of the RMCP's System Abstraction Layer. $\beta_{k,m}$ and $\pi_{k,m}$ are the RMCP processor selection and scheduling strategies for the kth priority class on the mth processor. $\mathcal{L}_m$ represents the set of L_m packets in the mth processor's queue.

of the RMCP. The RMCP assumes that video packets from H remote applications (i.e., video sources) are streamed, from a network (or storage device), into an input buffer present at the client system. Video packets are then pushed into M parallel queues that are served by M processors (with one queue per processor). The decoded packets from each processor are then recombined at the output buffer and the H sequences are simultaneously displayed for the client. In this paper, we do not consider the streaming delays or the pre-roll delays at the input buffer because we are primarily concerned with the delays incurred in the processing pipeline. Specifically, we consider the queuing delays and the processing delays. The main purpose of the SAL is to analyze (before and during run-time) the expected queuing and processing delays for the set of active video applications described by the AAL.

C. RMCP Strategies

The queuing delays incurred by different video packets depend on the deployed RMCP processor selection and real-time scheduling strategies. These strategies are the means through which the RMCP controls the system congestion (i.e., queuing delays) and, ultimately, the decoded video quality across all H sequences.

For every video packet l in the video decoding system shown in Fig. 2, many RMCP strategies exist for serving the packet. Let $\mathcal{S}_{k,m} = \mathcal{S}^{\pi}_{k,m} \times \mathcal{S}^{\beta}_{k,m}$ denote the set of RMCP strategies that can be deployed by class q_k packets on the mth processor. We denote $s_{k,m}(l \in q_k) = [\pi_{k,m}, \beta_{k,m}] \in S_{k,m}$ as the RMCP strategy selected for video packet l. The components of $s_{k,m}(l \in q_k) \in S_{k,m}$ are defined as follows.

- ***Packet scheduling strategy*** (Section V): The set $\mathcal{S}^{\pi}_{k,m}$ contains all possible scheduling strategies $\pi_{k,m}$ that can be deployed for a video packet $l \in q_k$ on the m th processor. The scheduling strategy determines which packets in the mth processor's queue are allocated processing time. The packet schedules depend on the decoding deadlines τ_k, distortion reductions Δd_k, and the DAG that represents the decoding dependencies $\mathrm{DAG}(\mathcal{L}^{(h)})$ $(1 \le k \le K, 1 \le h \le H)$.
- ***Processor selection strategy*** (Section VI): The processor selection parameters $\beta_{k,m} \in \mathcal{S}^{\beta}_{k,m}$ specify the fraction of class q_k video packets in the input buffer that will be forwarded to the mth processor's queue $(1 \le m \le M)$. That is, $\beta_{k,m}$ allows the decoding workload of class q_k packets to be distributed across the M processors. The processor selection parameters are defined such that $\sum_{m=1}^{M} \beta_{k,m} = 1$.

We note that, in Fig. 2, $\mathcal{L}_m$ represents the set of L_m packets in the mth processor's queue $(1 \le k \le K)$.

D. Problem Formulation

Given the video packet prioritization scheme, dependencies between different packets, and the RMCP strategy space, we desire to minimize the total distortion over the N_{inp} packets, from all H video sequences, that are currently in the input buffer. Note that $N_{\mathrm{inp}} = \sum_{k=1}^{K} N_k$, where N_k is the number of packets in class q_k. Let us define the joint RMCP strategy vector $\mathbf{s} = \{s_{k,m}(l \in q_k) \mid l = 1,\ldots,N_{\mathrm{inp}}, 1 \le k \le K, 1 \le m \le M\}$ as a vector of RMCP strategies that can be deployed for packets in the input buffer (i.e., $s_{k,m}(l \in q_k) \in \mathbf{s}$ specifies the lth packet's scheduling strategy and the processor on which it will be decoded). The problem of selecting the joint RMCP strategy vector $\mathbf{s}^{\mathrm{opt}}$ that minimizes the total distortion becomes

$$\mathbf{s}^{\mathrm{opt}} = \arg\min_{\mathbf{s}} \left(D_0 - \sum_{l=1}^{N_{\mathrm{inp}}} w_l(\mathbf{s}) \cdot \Delta d_l \right) = \arg\max_{\mathbf{s}} \sum_{l=1}^{N^{\mathrm{inp}}} w_l(\mathbf{s}) \cdot \Delta d_l \quad (3)$$

where D_0 is the distortion if no packets are decoded for any of the video sequences and $w_l(\mathbf{s})$ is an indicator variable that takes value 1 if the lth packet is successfully decoded based on the

deployed strategies $s_{k,m}(l \in q_k) \in \mathbf{s}$ and takes value 0 otherwise. Importantly, whether or not the lth packet is successfully decoded is dependent on its own strategy $s_{k,m}(l \in q_k)$ as well as the strategies selected for all of the other packets.

The above problem is intractable for several reasons. Firstly, even when we consider a single processor scenario (i.e., $M = 1$) the complexity of this algorithm is $O(N_{\text{inp}}!)$ which makes an exhaustive search for the optimal solution unreasonably complex (in fact, as will be discussed in Section V, just determining the optimal scheduling on a single processor is NP-complete). Secondly, we do not know *a priori* the actual decoding complexity of the packets. Hence, it is impossible to determine whether a packet l will be successfully decoded given the joint strategy vector $\mathbf{s}$ (i.e., we cannot determine $w_l(\mathbf{s})$ *a priori*).

To suboptimally solve the intractable problem in (3), we decompose it into two separate problems. In Section V, we simplify the packet scheduling problem (i.e., selecting $\pi_{k,m} \in \mathcal{S}^{\pi}_{k,m}$) by dividing it into M separate scheduling problems (one for each processor queue). In Section VI, we propose a processor selection algorithm (i.e., an algorithm for selecting $\beta_{k,m} \in \mathcal{S}^{\beta}_{k,m}$) that distributes the packets in the various priority classes to the M processor queues where they wait to be scheduled. This algorithm attempts to reduce decoding deadline expirations by minimizing the queue waiting times across the M processors.

III. Workload Distributions at the Application Abstraction Layer

The GOP structure and decoding dependencies described in Section II-A are particularly important because they dictate the decoding deadline distribution of frames within a GOP. The decoding deadlines, in turn, influence the system workload characteristics (i.e., the processor load over time) because they determine when each frame must be decoded. In this section, we analyze and model the workload/complexity distributions associated with decoding a single frame. This will facilitate our analysis of the expected queuing delays experienced by the video packets in the various priority classes,

I, P, and B frame types have widely varying complexity distributions that are dependent on many factors including the source characteristics (e.g., high-motion, complex textures, object occlusion) and the encoding parameters (e.g., the average bit rate or the quantization parameter [4]). In order to measure the frame decoding complexities, we profiled the JM 10.2 H.264/AVC reference software with system calls that query the internal processor counter to obtain high-resolution timing measurements. The measurements are taken on a Dell Pentium IV system running Microsoft Windows XP safe mode with only a command prompt to ensure that context switches and interrupting processes are minimized. The reported measurements for the decoding complexities of individual frames are taken as the minimum measurements over three trials. This technique is used to further minimize outlier measurements caused by context switches and interrupting processes that are not representative of the actual decoding complexity.

A video packet's decoding complexity is dependent on its frame type, source sequence (e.g., Mobile or Foreman), and quantization parameter which we denote as $fr(l) \in \{I, P, B\}$, seq, and QP, respectively. Based on our decoding complexity measurements, we found that the normalized complexity, $\hat{C}^{\text{seq}}_{fr(l),QP}$, for each frame type (for a particular source sequence encoded with a fixed quantization parameter) is a random variable with a distribution that is nearly Poisson, i.e.,

$$\hat{C}^{\text{seq}}_{fr(l),QP} \sim p^{\text{seq}}_{fr(l),QP}(n) = \frac{\left(\lambda^{\text{seq}}_{fr(l),QP}\right)^n \cdot e^{-\lambda^{\text{seq}}_{fr(l),QP}}}{n!} \quad (4)$$

where n is a Poisson bin number, $p^{\text{seq}}_{fr(l),QP}(n)$ is the probability that the normalized complexity falls into bin n, and $\lambda^{\text{seq}}_{fr(l),QP}$ is a shape parameter for the normalized complexity distribution. The actual decoding complexity, $C^{\text{seq}}_{fr(l),QP}$ (measured in processor cycles), can be modeled by a shifted and scaled Poisson distribution, as follows:

$$C^{\text{seq}}_{fr(l),QP} = a^{\text{seq}}_{fr(l),QP} \cdot \hat{C}^{\text{seq}}_{fr(l),QP} + b^{\text{seq}}_{fr(l),QP} \quad (5)$$

where $a^{\text{seq}}_{fr(l),QP}$ and $b^{\text{seq}}_{fr(l),QP}$ are constants (with units in processor cycles) that depend on the sequence, frame type, and quantization parameter. Table I shows example affine transform coefficients and shape parameters for the decoding complexity $C^{\text{seq}}_{fr(l),QP}$ when $fr(l) = I, P, B$, $QP = 39, 24$, and $\text{seq} = mob, for, ste, coa$. Note that, in Table I, the superscripts and subscripts on $a^{\text{seq}}_{fr(l),QP}$, $b^{\text{seq}}_{fr(l),QP}$, and $\lambda^{\text{seq}}_{fr(l),QP}$ are removed for simplicity of notation. In the Appendix, we describe how we derived the affine transform coefficients and shape parameters shown in Table I using histograms of our complexity measurements.

In a real-time streaming scenario, the complexity model parameters are not be known *a priori*. Therefore, these parameters need to be estimated at the decoder. To facilitate this estimation process, hinting information or metadata can be created at the encoder and transmitted with the bitstream [12]. This hinting information can be used by the decoder to estimate the affine transform coefficients and the shape parameters for the different frame types. More specifically, the hinting information can contain a feature vector that describes the sequence content characteristics such as the number of motion vectors, percentage of non-zero transform coefficients etc. and/or encoding parameters such as the quantization-parameter (QP). Based on the feature vectors communicated via the metadata, an online classifier [16] at the decoder can be used to determine the complexity model parameters for the I, B, and P frames. This online classifiers performance will depend on previous offline training or profiling [4], [12] done at the decoder on a variety of representative video sequences with different content characteristics and encoded with different encoding parameters (e.g., different values of QP).

IV. Queuing Delay Analysis at the System Abstraction Layer

In this section, we develop a model that the SAL can deploy in order to analyze the queuing delays experienced by packets in different priority classes in the M processor queues. Due to the delay-sensitive nature of the video applications, knowledge of the queue waiting times $W_{k,m}$ (i.e., the time that packets in priority class q_k spend in the mth processor's queue before they

TABLE I
AFFINE TRANSFORM COEFFICIENTS AND SHAPE PARAMETERS FOR THE DECODING COMPLEXITY OF THE I, P, AND B FRAMES IN THE MOBILE, FOREMAN, STEFAN, AND COASTGUARD SEQUENCES (CIF RESOLUTION, 30 HZ, QP 39 AND 24)

Frame Type	Mobile (CIF, 30 Hz, QP 39)			Mobile (CIF, 30 Hz, QP 24)			Foreman (CIF, 30 Hz, QP 39)			Foreman (CIF, 30 Hz, QP 24)		
	a	b	λ	a	b	λ	a	b	λ	a	b	λ
I	1.0e6	6.3e7	3.64	7.5e6	9.9e7	0.924	2.0e6	4.1e7	2.92	3.5e6	5.7e7	3.76
P	8.7e5	7.4e7	42.40	2.1e6	8.0e7	23.8	3.0e6	4.3e7	17.7	9.5e6	5.5e7	6.03
B	5.8e6	8.9e7	10.00	1.8e6	8.4e7	39.7	2.2e7	5.6e7	2.79	8.7e6	8.8e7	6.69
Frame Type	**Stefan** (CIF, 30 Hz, QP 39)			**Stefan** (CIF, 30 Hz, QP 24)			**Coastguard** (CIF, 30 Hz, QP 39)			**Coastguard** (CIF, 30 Hz, QP 24)		
	a	b	λ	a	b	λ	a	b	λ	a	b	λ
I	9.7e4	5.0e7	30.05	2.7e6	7.4e7	2.87	1.1e6	4.4e7	3.43	1.6e6	6.3e7	6.5
P	1.1e6	5.7e7	41.08	2.0e7	7.6e7	2.16	2.1e6	6.8e7	12.38	1.2e6	7.6e7	30.75
B	1.5e7	7.7e7	3.95	1.1e7	8.6e7	6.21	2.4e7	7.6e7	2.15	9.1e6	8.6e7	6.68

are served) is particularly important. Since $W_{k,m}$ is a random variable, the waiting time tail distribution $\Pr(W_{k,m} > t)$ gives the probability that the waiting time $W_{k,m}$ exceeds t seconds. When $t = \tau_k$, the waiting time tail distribution can be used to estimate the probability that class q_k packets will be successfully decoded (before their decoding deadline τ_k) given the deployed RMCP strategies described in Section II-C. In Section IV-A, we use queuing theory to determine $\Pr(W_{k,m} > t)$.

A. Priority Queuing Analysis

The SAL models the system using M/G/1 preemptive priority queues. Using this queuing model and the set of active video applications parameterized by the AAL, the SAL is able to make low-complexity estimates of the average system performance (i.e., expected waiting times $E[W_{k,m}]$ and tail distributions $\Pr(W_{k,m} > t)$) with only a few parameters.

Let η_k be the arrival rate of class q_k packets into the client system's input buffer illustrated in Fig. 2. η_k can be determined based on the network rate and coding structures or by observing the packet arrivals over time. Given the processor selection parameters, $\beta_{k,m}$, the arrival rate of class q_k packets into the Mth processor's queue is:

$$\eta_{k,m} = \beta_{k,m} \cdot \eta_k \tag{6}$$

Let $E[W_{k,m}]$ be the expected waiting time for class q_k packets in the mth processor queue. For a preemptive priority M/G/1 queue, the Pollaczek–Khinchin equation gives the following result [11]:

$$E[W_{k,m}] = \frac{\sum_{i=1}^{k} \eta_{i,m} E[X_i^2]}{2\left(1 - \sum_{i=1}^{k-1} \eta_{i,m} E[X_i]\right)\left(1 - \sum_{i=1}^{k} \eta_{i,m} E[X_i]\right)} \tag{7}$$

where the service time X_l is related to the workloads, C_l, parameterized by the AAL (see Section III) as

$$X_l = \frac{1}{f} \cdot C_l, \tag{8}$$

and f is the operating frequency of the processor on which packet l is decoded[2]. Based on $E[W_{k,m}]$, the tail distribution of the waiting time can be approximated as

$$\Pr(W_{k,m} > t) \approx \left(\sum_{i=1}^{k} \eta_{i,m} E[X_i]\right) \times \exp\left(-t \times \frac{\sum_{i=1}^{k} \eta_{i,m} E[X_i]}{E[W_{k,m}]}\right) \tag{9}$$

In (9), we adopt the G/G/1 tail distribution approximation based on the work of [18], [19]. Note that (7) and (9) assume that class q_k packets have lower priority than the packets in classes $q_1, \ldots, q_{k-1}$. Additionally, (7) requires that

$$\sum_{i=1}^{k} \eta_{i,m} E[X_i] < 1, \quad 1 \le k \le K. \tag{10}$$

If the inequality in (10) does not hold, then we can greedily reduce the arrival rate of the lower priority classes until the condition is met.

Before we can evaluate (7) and (9), we must evaluate the first and second moments of the service time distributions. Based on the normalized complexity distribution in (4), the complexity model definition in (5), and the conversion between the complexity distribution and service time distribution in (8), the first and second moments of the service time distributions can be expressed as

$$\begin{aligned} E[X_l] &= E\left[\frac{C_l}{f}\right] = \frac{1}{f}(a\lambda + b), \\ E[X_l^2] &= E\left[\left(\frac{C_l}{f}\right)^2\right] \\ &= \frac{1}{f^2}\left(a^2(\lambda + \lambda^2) + 2ab\lambda + b^2\right) \end{aligned} \tag{11}$$

where a, b, and λ are the affine transform coefficients and the shape parameter used in (4)–(5) with the superscripts and subscripts removed for notational simplicity.

[2]With dynamic-voltage-scaling (DVS) [17] enabled processors, the frequency f can vary. In this paper, we assume that f is a constant. Integrating DVS into the proposed framework is a topic of our future research.

B. Expected Decoded Distortion

The decoded video quality is influenced by many factors, e.g., source content characteristics, competing tasks, available processing power, workload traffic characteristics, the deployed RMCP strategies etc. All of these factors are not necessarily known a-priori and, in many cases, can only be estimated during the actual real-time decoding. Using the M/G/1 preemptive priority queuing system model, however, the RMCP is able to predict the expected decoded distortion $D_{\exp}(t_{\text{cur}})$ across all H active video decoding tasks based on the waiting-time tail distribution approximation in (9)

$$D_{\exp}(t_{\text{cur}}) = D_0 - \sum_{k=1}^{K}\sum_{m=1}^{M} \Delta d_k \cdot \beta_{k,m} \cdot \left(1 - \Pr(W_{k,m} > \tau_k - t_{\text{cur}} - E[X_k])\right). \quad (12)$$

where D_0 is the decoded distortion if no packets are decoded for any video sequence. $D_{\exp}(t_{\text{cur}})$ is an estimate of the expected (time-average) decoded distortion based on the set of active tasks and the system congestion at time t_{cur}. The expression for the distortion in (12) is similar to the expressions for the distortion used in Section II-D and (15) in Section V. However, (12) models the expected decoding distortion based on the probability that packets will be successfully decoded, whereas (3) and (15) express the actual distortion reduction given the successfully decoded packets (i.e., the packets for which $w_l = 1$).

V. Video Packet Scheduling

Consider a scenario where packets belonging to H different video sequences are present in the decoding system at time t_{cur}. Let $\mathcal{L}_m$ represent the set of L_m packets in the m th processor's queue. We assume that all expired packets are purged from the queue; therefore, if $l \in \mathcal{L}_m$, then $t_{\text{cur}} < \tau_l$. Given the set of packets, $\mathcal{L}_m$, we desire to determine which packets should be decoded based on some scheduling policy π, i.e., π selects a packet $l^* \in \mathcal{L}_m$ to decode. We write

$$l^* = \pi(\mathcal{L}_m, \mathbf{DAG}) \quad (13)$$

to denote that π selects the packet l^* from the set $\mathcal{L}_m$. In (13), $\mathbf{DAG}$ denotes the decoding dependencies of all H video sequences in the decoding system, i.e.,

$$\mathbf{DAG} = \{\text{DAG}(\mathcal{L}^{(h)}) \mid 1 \le h \le H\}. \quad (14)$$

The packet scheduling depends on the sequences' decoding dependencies because it is subject to the constraint that the packet $l^* = \pi(\mathcal{L}_m, \mathbf{DAG})$ has had all of its decoding dependencies fulfilled, i.e., $w_{l'} = 1, \forall\, l' \in A_{l^*}$. For the set of packets $\mathcal{L}_m$ in the mth processor's queue the optimal scheduling vector $\pi^{\text{opt}}(\mathcal{L}_m, \mathbf{DAG}) = (\pi_1^{\text{opt}}, \ldots, \pi_{L_m}^{\text{opt}}) \in \Pi$, with π_l^{opt} representing the scheduling decision for the lth packet, is a vector of scheduling decisions that maximizes the distortion reduction over the set of all possible scheduling decision vectors, Π. Formally

$$\pi^{\text{opt}}(\mathcal{L}_m, \mathbf{DAG}) = \arg\max_{\pi\in\Pi} \left\{ \sum_{l\in\mathcal{L}_m} w_l(\pi_l) \cdot \Delta d_l \right\} \quad (15)$$

where $w_l(\pi_l)$ is an indicator function that takes value 1 if the scheduling decision π_l results in successful decoding of the lth packet, and takes value 0 otherwise. This scheduling problem belongs to a family of scheduling problems that are known to be NP-complete and therefore it cannot be solved in polynomial time [13], [15]. In the rest of this section, we present several sub-optimal heuristic scheduling policies that aim to maximize the distortion reduction as in (15). Unlike the schedule determined in (15), the heuristic scheduling policies schedule a single packet whenever a processor finishes processing the previously scheduled packet.

In Section V-A, we describe a conventional deadline-driven scheduling algorithm based on an EDF discipline. In Section V-B, we describe a priority-driven scheduling policy that schedules packets in descending order of their distortion-reduction parameter. We observe that, in the explored multitask video decoding scenario using H.264/AVC compliant bit streams, both of these policies have shortcomings that can result in unnecessary packet deadline expirations and, consequently, quality degradation. In Section V-C, we introduce a risk-aware scheduling policy that jointly considers the distortion impacts and delay deadlines of the packets $l \in \mathcal{L}_m$, and in Section V-D we provide an illustrative example comparing the three policies. Finally, in Section V-E, we propose an additional scheduling policy that makes scheduling decisions by "looking-ahead" to estimate the consequences (in terms of potential distortion reduction) of the risk-aware scheduling decision.

A. Earliest-Deadline-First Scheduling

A frequently used scheduling strategy for multimedia systems is the EDF policy [6] in which the decodable packet with the earliest deadline is the first packet to be allocating processing resources. The EDF scheduling policy can be written as

$$\begin{aligned} l^{(\text{EDF})} &= \pi_{\text{EDF}}(\mathcal{L}_m, \mathbf{DAG}) \\ &= \arg\min_{l\in\mathcal{L}_m} \{\tau_l \mid w_{l'} = 1, \forall\, l' \in A_l\} \end{aligned} \quad (16)$$

where the superscript (EDF) indicates that the packet was selected by the EDF scheduling policy.

It is clear from (16) that π_{EDF} does not consider the decoded video quality. Hence, as illustrated by the following example, the EDF policy will perform sub-optimally in terms of achieved video quality when the system is heavily loaded. Consider a scenario where frames of a video sequence encoded with an *IBBPBB* GOP structure are streamed into a single processor queue. At the GOP boundary, just before the next I frame with deadline $\tau^{(I)}$ and distortion impact $\Delta d^{(I)}$ is to be decoded, a B frame with an earlier deadline $\tau^{(B)}$ and lower distortion impact $\Delta d^{(B)}$ waits to be decoded. Since $\tau^{(B)} < \tau^{(I)}$, the EDF scheduling policy will decode the B frame packet before the I frame packet. If the B frame's service time is too long because of the system's limited processing capabilities or the time-varying decoding complexity, then the I frame may not be successfully decoded before its deadline. Consequently, none of the frames in the subsequent GOP that depend on this I frame can be decoded.

In Section V-D, we provide an example that highlights the EDF policy's limitations when two video decoding tasks are competing for the same processor resources.

B. Priority-Driven Scheduling

Alternatively, we can deploy a scheduling strategy based on a PQ policy in which the decodable packet with the highest distortion impact is the first packet to be allocated processor time. The PQ scheduling policy may be written as

$$\begin{aligned} l^{(\mathrm{PQ})} &= \pi_{\mathrm{PQ}}(\mathcal{L}_m, \mathbf{DAG}) \\ &= \arg\max_{l \in \mathcal{L}_m} \{\Delta d_l \mid w_{l'} = 1,\ \forall\, l' \in A_l\}. \end{aligned} \tag{17}$$

Although π_{PQ} directly considers the instantaneous distortion impact when scheduling a packet, it does not consider the relative decoding deadlines of the packets in the mth processor's queue. For instance, consider again the scenario where frames of a video sequence encoded with an *IBBPBBPBB* GOP structure are streamed into a single processor queue. After decoding the first P frame in a GOP, it is possible to decode the subsequent P frame or B frame. However, due to the dependencies between the frames, $\Delta d^{(P)} > \Delta d^{(B)}$. Therefore, under the PQ policy, the P frame will be decoded before the B frame even if there is enough time to first decode the B frame and then the P frame. Unfortunately, after decoding the P frame well before its deadline it is unlikely that the B frame can be processed before its deadline expires.

In Section V-D, we underscore the PQ policy's limitations in a scenario where two video decoding tasks compete for the same processing resources.

C. Risk-Aware Scheduling

Based on the above observations related to the deadline-driven EDF and priority-driven PQ scheduling policies, it is clear that a scheduling policy that simultaneously considers the priorities and decoding deadlines of the various packets in $\mathcal{L}_m$ is desirable. We consider a risk-aware scheduling policy where the lth packet's *risk*, denoted as θ_l^{risk}, is defined as its decoding distortion weighted by the instantaneous, decoding slack dependent, probability that it will not be decoded in time. Letting t_{cur} denote the current time and $E[X_l]$ denote the expected service time for the lth packet, the expected decoding slack (i.e., the expected time remaining before a packet's deadline after the packet is decoded) can be expressed as

$$E[\tau_l^{\mathrm{slack}}] = \tau_l - t_{\mathrm{cur}} - E[X_l]. \tag{18}$$

This definition of slack is congruent with that used in real-time system scheduling literature [20]. Using (18), we define the lth packet's risk as:

$$\theta_l^{\mathrm{risk}} = \begin{cases} \Delta d_l \cdot \Pr(W_l > E\left[\tau_l^{\mathrm{slack}}\right]), & \text{if } E\left[\tau_l^{\mathrm{slack}}\right] > 0 \\ 0, & \text{if } E\left[\tau_l^{\mathrm{slack}}\right] \le 0 \end{cases} \tag{19}$$

where we set $\theta_l^{\mathrm{risk}} = 0$ when $E[\tau_l^{\mathrm{slack}}] \le 0$ (i.e., $t_{\mathrm{cur}} + E[X_l] > \tau_l$) because packets that meet this condition are, on average, unlikely to be successfully decoded. The risk-aware scheduling policy, which schedules the decodable packet with the highest risk, can be expressed as:

$$\begin{aligned} l^{(\mathrm{RA})} &= \pi_{\mathrm{RA}}(\mathcal{L}_m, \mathbf{DAG}) \\ &= \arg\max_{l \in \mathcal{L}_m} \left\{\theta_l^{\mathrm{risk}} \mid w_{l'} = 1,\ \forall\, l' \in A_l\right\}. \end{aligned} \tag{20}$$

In general, the RA scheduling policy attempts to schedule packets with more imminent deadlines unless a packet with a greater distortion impact is at risk of violating its own decoding deadline. The benefits of the RA scheduling policy, compared to the EDF and PQ policies, are exemplified in Section V-D.

D. Scheduling Policy Comparison

A simple example illustrating the three scheduling policies described in Sections V-A–C is shown in Fig. 3(a)–(c). Fig. 3(a) shows a set of four packets in a congested queue where packets l =1, 3 belong to sequence h and packets l =2, 4 belong to another sequence, h'. The decoding dependencies, expected service times, distortion reductions, and decoding deadlines are shown for each packet. We say the queue is congested because the expected time required to empty the queue (i.e., 100 ms) is greater than the total amount of time available to decode all of the packets before their deadlines (i.e., packet $l = 4$ has the latest decoding deadline at $\tau_4 = 80$ ms). Hence, some packets will not be decoded regardless of the deployed scheduling policy. Importantly, in the context of video decoding and multimedia systems, the scheduling policy cannot be evaluated by the number of packets that are successfully decoded. Rather, the scheduling policy should be evaluated based on the average distortion reduction achieved under the deployed policy.

Fig. 3(c) demonstrates the three packet scheduling policies based on the set of packets and their corresponding parameters shown in Fig. 3(a). To simplify the example in Fig. 3(c), we assume that each packet requires exactly its expected service time to be decoded. The total distortion reduction achieved by each scheduling strategy is shown at the right of the figure and is calculated as $\sum_{l=1}^{4} w_l \cdot \Delta d_l$, where w_l indicates whether or not packet l is successfully decoded. It is clear from Fig. 3(c) that π_{EDF} [see (16)] performs inadequately in terms of overall distortion reduction. In fact, in this example, π_{EDF} selects the least important packets for decoding. The first packet selected by π_{EDF} (i.e., $l = 1$) not only contributes very little to the overall quality, but the time spent decoding it makes it impossible to decode the high priority packet $l = 2$ prior to its deadline. Consequently, high priority packet $l = 4$, with the set of ancestors $A_4 = \{2\}$, cannot be decoded despite there having been enough processor idle time to have served it. The total distortion reduction using the EDF scheduling policy is $\sum_{l=1}^{4} w_l \cdot \Delta d_l = 15$, in this example.

Observing Fig. 3(c) again, we see that π_{PQ} decodes the two highest priority packets. However, by decoding packet $l = 4$ well before its deadline, packet $l = 3$ cannot be decoded before its deadline expires. The total distortion reduction using the PQ scheduling policy, in this example, is $\sum_{l=1}^{4} w_l \cdot \Delta d_l = 35$.

Finally, Fig. 3(b)–(c) illustrates the risk-aware scheduling policy, π_{RA} [see (20)]. Fig. 3(b) demonstrates how the risk, θ_l^{risk} [see (19)], changes over time for packets l =1, 2, 3, 4.

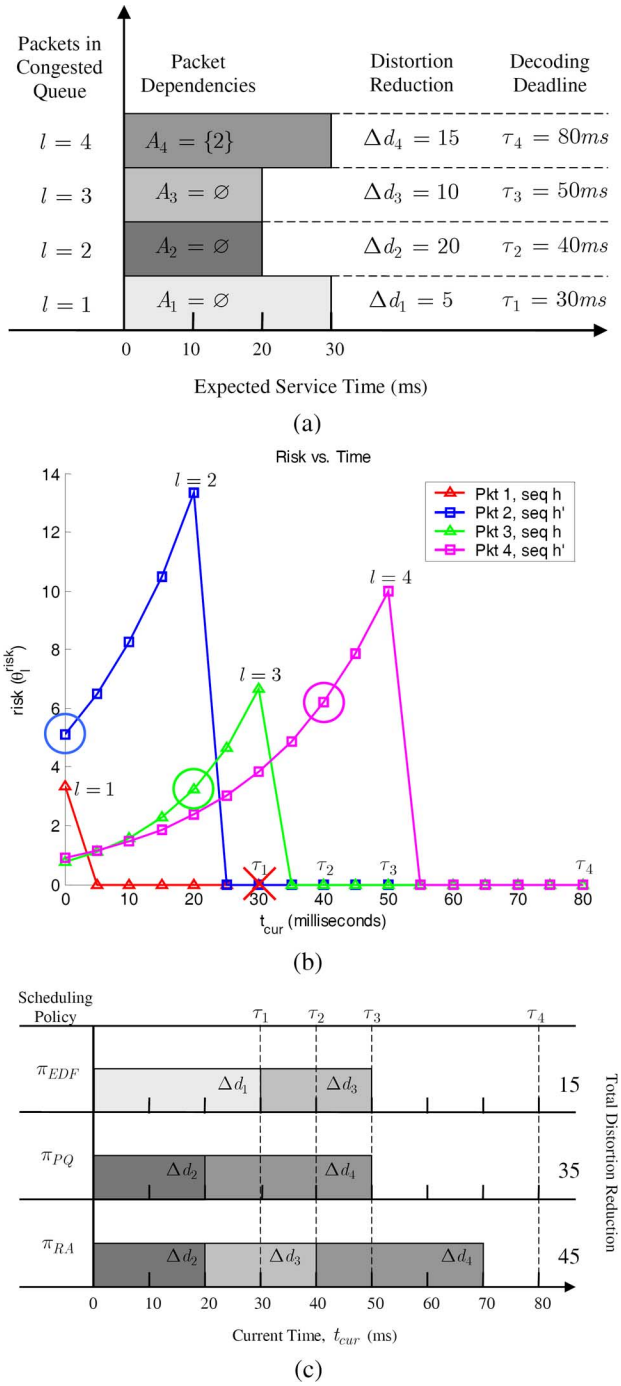

Fig. 3. (a) Packets in the congested queue. Packet dependencies, expected service times, distortion reductions, and decoding deadlines. (b) Risk-aware scheduling showing θ_l^{risk} versus t_{cur}. (c) Packet scheduling based on the earliest-deadline-first ((π_{EDF})), Priority-Queuing (π_{PQ}), and risk-aware (π_{RA}) scheduling policies. The shaded boxes correspond directly to the packets in Fig. 3(a) and the risk-aware scheduling is based on the risk curves in Fig. 3(b).

In Fig. 3(b), the large rings indicate the time at which the corresponding packet begins service. Immediately after a packet is decoded, the next packet is scheduled for service based on π_{RA}. The "×" indicates that packet $l = 1$ is discarded at $t_{\text{cur}} = 30$ ms $= \tau_1$ because its decoding deadline is expired. In this example, the total distortion reduction using the RA scheduling policy is $\sum_{l=1}^{4} w_l \cdot \Delta d_l = 45$. The RA policy outperforms the EDF and PQ policies in this example because it considers the distortion reduction in conjunction with the temporal proximity of the packet deadlines, thereby enabling it to make more dynamic scheduling decisions.

E. Proposed Look-Ahead Risk-Aware Scheduling

A weakness of the RA scheduling policy is that it is sensitive to many parameters such as the number of priority classes, the packet arrival rates, the first and second moments of the service time distributions etc. Consequently, unless the applications are carefully controlled, the RA policy can yield erratic results. In this subsection, we augment the RA policy to stabilize its behavior by considering the instantaneous risk estimations in (19) and the *future* schedulability and distortion impact of the packets in $\mathcal{L}_m$. To this end, we propose a look-ahead risk-aware (LR) scheduling policy that, at the current time t_{cur}, considers the RA scheduling decision $l^{(\text{RA})} = \pi_{\text{RA}}(\mathcal{L}_m, \mathbf{DAG})$ and then evaluates the potential distortion reduction at the time $t_{\text{cur}} + E[X_{l^{(\text{RA})}}]$. If an alternate scheduling decision is found that can achieve a greater distortion reduction, this scheduling decision is made at time t_{cur} instead of the initially considered RA scheduling decision.

The following describes the LR scheduling algorithm pseudocode shown in Table II.

- ***Initialization (Steps 1–3):*** When the mth processor becomes idle at time t_{cur} (step 1), we determine the packet $l^{(\text{RA})} = p_{\text{RA}}(\mathcal{L}_m, \mathbf{DAG})$ that would be scheduled by the RA policy (step 2). We initialize the variable *count* to 0 (step 3). *count* is used to record the number of packets $l \in \mathcal{L}_m \backslash \{l^{(\text{RA})}\}$ that are expected to be unschedulable after $l^{(\text{RA})}$ is serviced and that have a greater distortion impact than $l^{(\text{RA})}$.
- ***Look-ahead and evaluate alternative scheduling options (Steps 4–8):*** For all packets $l \in \mathcal{L}_m \backslash \{l^{(\text{RA})}\}$ with their decoding dependencies satisfied (step 4), we look-ahead to the time $t_{\text{cur}} + E[X_{l^{(\text{RA})}}]$ to estimate the distortion impact of scheduling $l^{(\text{RA})} = \pi_{\text{RA}}(\mathcal{L}_m, \mathbf{DAG})$ (steps 5–8). If $E[\tau_l^{\text{slack}}] - E[X_{l^{(\text{RA})}}] \leq 0$ (i.e., it is expected that there will be no time remaining to decode packet l after decoding $l^{(\text{RA})}$) and $\Delta d_l > \Delta d_{l^{(\text{RA})}}$ (i.e., decoding packet l instead of $l^{(\text{RA})}$ will result in greater distortion reduction), then we record Δd_l (step 7) and increment `count` by one (step 8).
- ***Schedule packet to maximize the distortion reduction over time (Steps 9–12):*** If `count` is non-zero (step 9), then we schedule packet $l = \pi_{\text{LR}}(\mathcal{L}_m, \mathbf{DAG})$ that maximizes Δd_l (step 10). If `count` is 0 (step 11), then we schedule packet $l^{(\text{RA})} = \pi_{\text{RA}}(\mathcal{L}_m, \mathbf{DAG}) = \pi_{\text{LR}}(\mathcal{L}_m, \mathbf{DAG})$ (step 12).

Similar to the RA scheduling policy, the LR policy attempts to schedule packets with more imminent deadlines unless a packet with a greater distortion impact is at risk of violating its own decoding deadline. The difference between the two policies is that the RA policy evaluates an instantaneous risk using (19) based on the priority queuing model and the packet's expected slack whereas the LR policy also considers the future distortion impact of the scheduling decision made at time t_{cur} using the expected service time for $l^{(\text{RA})}$ and the expected slack of the packets in $\mathcal{L}_m$.

Unlike the EDF and PQ policies, which are based on incomplete application and system models, π_{LR} integrates all of the

application model parameters (i.e., distortion impacts, deadline distributions, decoding dependencies, and workload distributions) at the AAL as well as the priority queuing system model (i.e., the waiting time and service time distributions) at the SAL to enable the RMCP to significantly improve the system performance in terms of the total decoded video distortion.

F. Complexity of the Look-Ahead Risk-Aware Scheduling Algorithm

As described in Section V-E, the LR scheduling algorithm requires: (i) determining the RA scheduling solution and (ii) evaluating if another scheduling decision would perform better in terms of the distortion impact. Both steps (i) and (ii) have complexity that increases linearly with the average number of packets in each queue. Hence, if each queue contains an average of $\bar{L}$ packets and there are a total of M queues, then the total complexity of an exhaustive search is $O(2M\bar{L})$. Since the scheduling decisions at each processor queue are determined independently, these overheads are amortized over all of the processors such that the complexity of an exhaustive search on each processor is $O(2\bar{L})$. We note that the queue can be very long and that the above analysis is for an *exhaustive* search. In practice, however, it is not necessary to scan through packets from more than a few frames due to their delay deadlines and inter-dependencies. The number of packets/frames that are considered can be a design parameter used to make tradeoffs between complexity and performance [15].

When determining the RA scheduling decision, each packet's risk must be calculated using (19). For packet l in priority class q_k, the number of floating point operations (FLOPs) required to calculate θ_l^{risk} is approximately 10·k (assuming that $E[X_l^2]$ and $E[X_l]$ are already calculated). The number of FLOPs is proportional to the priority class index k because the risk calculation depends on the G/G/1 tail distribution approximation in (9) and the expected waiting time of the preemptive priority M/G/1 queue in (7). Importantly, the number of FLOPs can be reduced to a small constant by pre-calculating all of the time-independent quantities in (7) and (9) and storing them in a lookup table.

VI. Processor Selection and Load Balancing

Depending on the set of active video applications, their coding structures, their decoding dependencies, deadline, and their workload distributions, the packets in the various priority classes should be pushed onto the processor queues according to different distributions of the processor selection parameters $\beta_{k,m} \in \mathcal{S}_{k,m}^{\beta}$, with $1 \le k \le K$ and $1 \le m \le M$ in order to minimize queuing delays and maximize the video quality. We propose a processor selection policy that attempts to minimize the total delay experienced by packets in each priority class from the time the packets $l \in q_k$ enter the processor queues to the time they start being served. This delay is estimated by the RMCP at the SAL using the M/G/1 preemptive priority queuing analysis. Let T_k denote the (random variable) queuing delay incurred by class q_k video packets over all the processor queues. The expected delay $E[T_k]$ for class q_k can be expressed as a function of the expected waiting times $E[W_{k,m}(\beta_{k,m})]$ which depend on the processor selection parameters $\beta_{k,m} \in \mathcal{S}_{k,m}^{\beta}$. The minimum expected delay is determined as

$$E[T_k] = \min_{\beta_{k,m},\ \forall\ m\in\{1,\ldots,M\}} \left(\frac{1}{M} \sum_{m=1}^{M} E[W_{k,m}(b_{k,m})] \right). \tag{21}$$

If we minimize $E[T_k]$, then the average decoding slack $\tau_k - t_{\text{cur}} - E[T_k]$ is maximized. Observing the expression for the expected decoded distortion in (12), it is clear that maximizing the decoding slack allows us to obtain a lower distortion because it is less likely that the waiting time will exceed a larger delay (i.e., $\Pr(W_{k,m} > t)$ decreases as the time t increases).

To determine the processor selection parameters, $\beta_{k,m}$, we employ a discouragement policy inspired by the balking arrival probability in queuing theory [21]. Based on the expected waiting time in the mth processor's queue, the discouragement policy sets a probability $(1 - \beta_{k,m})$ that class q_k video packets will not join the queue as follows:

$$\beta_{k,m} = \frac{\xi_k}{1 + \kappa \cdot E[W_{k,m}]^{\varphi}} \tag{22}$$

where κ and φ are constants. k is set depending on the arrival rate as in [22]. φ is set so that the discouragement policy favors the processor queue with the smallest waiting time and least congestion. Finally, ξ_k normalizes the processor selection parameters to ensure that $\sum_{m=1}^{M} \beta_{k,m} = 1$

$$\xi_k = \left(\sum_{m=1}^{M} \frac{1}{1 + \kappa \cdot E[W_{k,m}]^{\varphi}} \right)^{-1}. \tag{23}$$

where Table III summarizes the processor selection adaptation algorithm.

VII. Results

In this section, we evaluate the real-time system scheduling policies proposed in Section V in conjunction with the priority queuing framework proposed in Section IV-A. We present these results in terms of the PSNR which is commonly used to measure video quality. We also evaluate the processor selection algorithm proposed in Section VI. All of the experiments are done by simulation using actual complexity traces that we obtained in the same manner as the data collected for the workload traffic models (as described in Section III).

A. Scheduling With Competing Tasks

In this subsection, we explore the dynamics of the various scheduling strategies in a scenario where multiple video decoding tasks compete for a single processor's resources. For the simulations in this subsection, we set the processor frequency to 3.2 GHz, such that a single task can be decoded (in real-time) with only a few missed packet deadlines, but two simultaneous video decoding tasks will incur significant quality degradation.

Fig. 4(a) illustrates the distribution of the number of undecoded packets per priority class when the Mobile sequence (QP 24, GOP structure *IBBPBBPBBPBB*) has sole access to the processing resources. Recall that classes $q_1, \ldots, q_K$ are defined such that $\Delta d_1 \ge \cdots \ge \Delta d_K$, hence, undecoded packets in

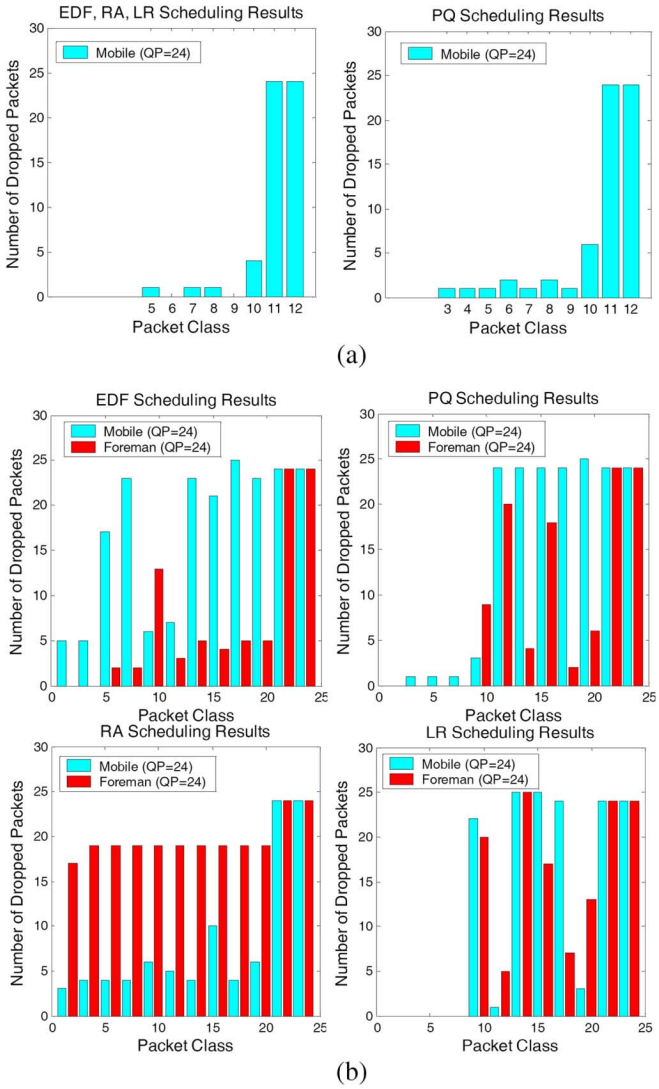

Fig. 4. Number of dropped packets per priority class for the EDF, PQ, RA, and LR scheduling policies. (a) Mobile sequence has sole access to the processor. (b) Mobile and Foreman sequences have to compete for processor resources.

higher priority classes result in higher distortion penalties. The left hand side of Fig. 4(a) illustrates the packet losses for the EDF, RA, and LR scheduling policies which, in this example of an uncongested system, all perform equivalently. This result exemplifies the power of the RA and LR scheduling strategies that are able to automatically adapt their behavior based on the system state. For example, when the system is uncongested (as in this scenario) the RA and LR policies behave more like the EDF policy. This is because, based on the queuing-theoretic model and the definition of the risk, the RA and LR policies can predict that the packets with the earliest deadline should be decoded (i.e., the risk is greatest for the packets with the earliest deadlines). Moreover, we will soon illustrate that the LR policy adjusts itself to behave more like the PQ policy in highly congested scenarios where there are insufficient resources to decode the lower priority packets.

Although the EDF policy performs quite well when the system is uncongested, the right-hand side of Fig. 4(a) illustrates that the PQ policy loses more high priority packets than the EDF, RA, and LR scheduling policies when the system is uncongested. This result appears counterintuitive. The PQ policy performs poorly in uncongested scenarios, however, because it will aggressively decode future I frames before decoding P frames with more imminent deadlines, thereby resulting in the expiration of the P frames' decoding deadlines. Consequently, the P frames cannot contribute to the decoded video quality nor can the subsequent B frames that depend on them.

Fig. 4(b) illustrates the distribution of the number of undecoded packets over the priority classes when simultaneously decoding the Mobile and Foreman (both with QP 24 and GOP structure*IBBPBBPBBPBB*) sequences on a single processor. Note that the priority classes for the Mobile sequence are shaded in the same way as they are in Fig. 4(b), however, the actual class assignments are different due to the additional classes comprising the Foreman sequence. With two tasks sharing the processing resources, the processor is significantly congested [which is clear when observing the large number of undecoded packets in Fig. 4(b)]. Therefore, heavy packet losses are inevitable. Depending on the deployed scheduling strategy, however, the more important packets (in terms of distortion impact) can be decoded while the packet deadline expirations are mitigated to the lower priority packets.

Fig. 4(b) shows that the EDF policy loses many significant frames including several of Mobile's I frames. The PQ policy, on the other hand, mitigates the packet expirations more effectively (by aggressively decoding the I and P frames) resulting in only a few dropped P frames from the Mobile sequence and no dropped I frames.

The results for the RA policy illustrate the sensitivity of the risk metric in [19]. The RA policy's poor performance is due to its sensitivity to myriad factors such as the expected distortion of the various packets, their delay deadlines, their expected service times, and the second moments of their service times as well as the arrival rates and expected waiting times of the priority classes. The RA scheduling results in Fig. 4(b) illustrate how the policy's sensitivity to these parameters make it unsuitable for multitask scheduling. Specifically, we observe that the RA policy results in the Mobile sequence greedily taking the processor resources from the Foreman sequence without consideration of the total decoded video quality across both sequences (i.e., many low priority packets from the Mobile sequence are decoded at the expense of the higher priority packets from the Foreman sequence).

Finally, Fig. 4(b) shows that the LR policy outperforms all the other scheduling policies by mitigating the packet losses to the lower priority classes and thereby successfully decoding more of the high priority packets and obtaining a better total video quality. We note that, in the congested two task scenario, the LR policy performs most like the PQ policy because it generally decodes the high priority packets before the low priority packets. However, the LR policy outperforms the PQ policy in this scenario because, if it detects that there is enough time to decode a lower priority packet without causing the deadline expiration of a higher priority packet, then it allocates processing resources to the lower priority packet. In this way, the LR policy avoids the pitfalls associated with the PQ policy's overly aggressive priority-driven scheduling decisions.

TABLE II
LOOK-AHEAD RISK-AWARE (LR) SCHEDULING ALGORITHM

1.	At time t_{cur} when processor m just becomes idle
2.	Determine $l^{(RA)} = \pi_{RA}(\mathcal{L}_m, \mathbf{DAG})$ by (20)
3.	// Record the *number of packets with* $E[\tau_l^{slack}] - E[X_{l^{(RA)}}] \le 0$ and $\Delta d_l > \Delta d_{l^{(RA)}}$ Initialize *count = 0*
4.	For all $l \in \mathcal{L}_m \setminus \{l^{(RA)}\}$ such that $\omega_{l'} = 1,\ \forall l' \in A_l$
5.	If $E[\tau_l^{slack}] - E[X_{l^{(RA)}}] \le 0$ // l *cannot be scheduled after* $l^{(RA)}$
6.	If $\Delta d_l > \Delta d_{l^{(RA)}}$
7.	Record Δd_l. // *Step 10 schedules packet* l *that maximizes* Δd_l
8.	*count* $\leftarrow$ *count + 1*
9.	If *count* $>$ *0* // $l^{(RA)}$ *did not maximize the expected distortion reduction*
10.	Schedule $l = \pi_{LR}(\mathcal{L}_m, \mathbf{DAG})$ to maximize the distortion reduction Δd_l
11.	Else // $l^{(RA)}$ *maximized the expected distortion reduction*
12.	Schedule $l^{(RA)} = \pi_{RA}(\mathcal{L}_m, \mathbf{DAG}) = \pi_{LR}(\mathcal{L}_m, \mathbf{DAG})$

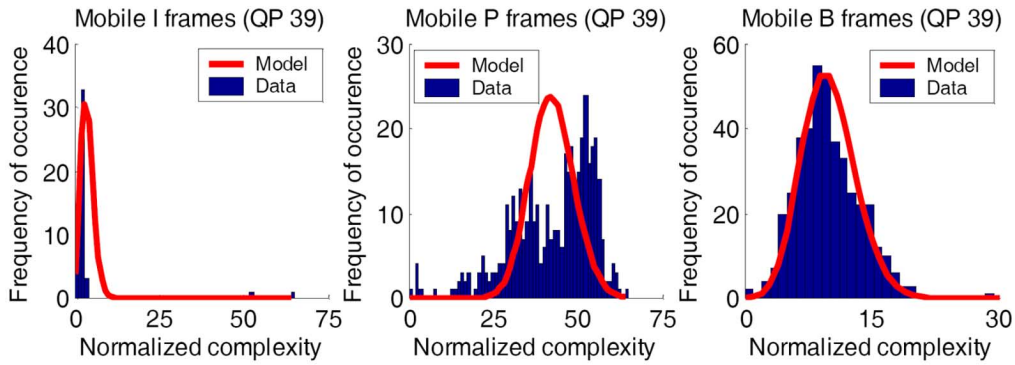

Fig. 5. Normalized complexity distributions.

TABLE III
PROCESSOR SELECTION ADAPTATION ALGORITHM

1.	Initialization: Set $\beta_{k,m} = 1/M$, $\forall k \in \{1,2,\ldots,K\}$, $\forall m \in \{1,2,\ldots,M\}$
2.	For each GOP
3.	For each priority class $k \in \{1,2,\ldots,K\}$ in descending order of Δd_k
4.	Determine $\beta_{k,m}$ using (22) and (23)
5.	Push packets onto processor queues according to $\beta_{k,m}$

Table IV illustrates the PSNR improvement of the PQ, RA, and LR policies relative to the EDF policy for the single task and two-task scenarios illustrated in Fig. 4. The bottom row of Table IV is the sum of the PSNR improvements of the Mobile and Foreman sequences in the two-task scenario. Note that, in the two task scenario, the LR policy performs 0.9 dB worse than the EDF policy when decoding Foreman (QP 24). However, the total decoded quality is highest (4.5 dB above the EDF policy) when the LR policy is deployed.

As we have illustrated, the PQ and LR policies perform better than the EDF policy when two video decoding tasks compete for a single processor's congested resources. However, we have found through extensive experiments (omitted here due to space constraints) that when a single task is using a congested pro-

TABLE IV
PSNR IMPROVEMENT OF THE PQ, RA AND LR SCHEDULING POLICIES RELATIVE TO THE EDF SCHEDULING POLICY BEFORE AND AFTER A SECOND TASK ENTERS THE DECODING SYSTEM AND COMPETES FOR PROCESSOR RESOURCES. NEGATIVE VALUES INDICATED WORSE PERFORMANCE THAN THE EDF POLICY

Number of Tasks	Sequence	Encoded PSNR	EDF Performance	PSNR Improvement For Scheduling Policy		
				PQ	RA	LR
Single task	Mobile (QP 24)	36.6 dB	34.4 dB	-1.3 dB	0.0 dB	0.0 dB
Two Competing Tasks	Mobile (QP 24)	36.6 dB	24.4 dB	4.2 dB	3.8 dB	5.4 dB
	Foreman (QP 24)	38.8 dB	32.4 dB	0.1 dB	-8.8 dB	-0.9 dB
	Total PSNR Improvement for Two Competing Tasks			4.3 dB	-5.0 dB	4.5 dB

TABLE V
AVERAGE QUEUING DELAY (ms) ACROSS TWO PROCESSORS AND THE CORRESPONDING ACHIEVED PSNR FOR THE MEDIA-AGNOSTIC AND MEDIA-AWARE SYSTEM DESIGNS

System Design	Priority Classes – Average Queueing Delay (ms)							
	q_1	q_2	q_3	q_4	q_5	q_6	q_7	q_8
Media-agnostic	184	164	173	166	476	372	1148	372
Media-aware	158	188	129	180	132	172	135	146

System Design	Sequence – Achieved PSNR (dB)		
	Mobile (QP 24)	Foreman (QP 24)	Total
Media-agnostic	33.9 dB	35.6 dB	69.5 dB
Media-aware	35.5 dB	36.9 dB	72.4 dB

cessor's resources, the PQ and LR policies[3] outperform the EDF policy by only a small margin on average (i.e., by up to 1.0 dB). This is because there are many less packets that can be decoded at any given time (due to the dependencies among packets) in single task scenarios. Therefore, the range of scheduling decisions is often limited to picking the highest priority packet or the packet with the earliest deadline. When multiple tasks compete for the same limited resources, however, more dynamic scheduling possibilities exist and it becomes more important to allocate the processor resources intelligently, based on a delay- and distortion-aware scheduling policy, to achieve the best video quality.

We note that, in the two task scenario in Table IV, the processor queue contained an average of 17.8 packets. Hence, based on the discussion on the complexity of the look-ahead risk-aware scheduling algorithm in Section V-F, it is clear that the scheduling incurs negligible processing overhead compared to the actual video decoding complexity in this example.

B. Combining Processor Selection and Scheduling

In this subsection, we evaluate the performance of the system illustrated in Fig. 2 when there are two video decoding tasks (Mobile with QP 24 and Foreman with QP 24) sharing resources on $M = 2$ parallel processors each with a 3.0 Ghz processor frequency. We compare a media-agnostic system design with a media-aware system design.

- **In the media agnostic scenario**, half of the packets from the two video sequences are arbitrarily (i.e without consideration of the priority classes, service times etc.) pushed onto each processor's queue. Subsequently, the packets are allocated processor resources using the EDF scheduling policy.
- **In the media-aware scenario**, video packets are pushed onto the processor queues using the processor selection scheme proposed in Section VI. Then, the packets in the queues are allocated processor time using the LR scheduling policy.

Our experimental results, illustrating the average queuing delay across the two processors (see [21]) and the achieved PSNRs in the media-agnostic and media-aware system scenarios, are shown in Table V. Although the EDF scheduling policy does not consider the video packet's priority class, we present the average queuing delays of packets in different priority classes to facilitate comparison between the media-agnostic and media-aware system designs. In Table V, only the first eight priority classes are shown because the decoded video quality depends most heavily on the delays experienced by packets in these classes. The system that deploys media-aware processor selection and scheduling strategies has significantly lower average queuing delays across almost all of the priority classes. The media-agnostic system, however, has some excessively large average queuing delays (i.e., > 1000 ms). These large delays indicate that many packets within that class had their deadlines expire before being successfully decoded. Such excessive delays are avoided in the media-aware scenario because the application specific queuing model, in conjunction with the processor selection scheme, anticipate and avoid resource bottlenecks.

VIII. CONCLUSION

In this paper, we propose a cross-layer design for video decoding systems that simultaneously decode multiple video streams on multiple parallel processors. One novelty of our

[3]In the single tast scenario, the RA also outperforms the EDF policy, on average, by approximately 0.5–0.1 dB. We omit the RA policy in our discussion here because it does not perform well under congested multitask scenarios.

TABLE VI
NUMBER OF SAMPLES AND SAMPLE VARIANCES FOR I, P, AND B FRAME SERVICE TIME MEASUREMENTS

Frame Type	Samples	Sample Variance			
		Mobile (CIF, 30 Hz, QP 39)	Mobile (CIF, 30 Hz, QP 24)	Foreman (CIF, 30 Hz, QP 39)	Foreman (CIF, 30 Hz, QP 24)
I	69	2.84e-5	8.97e-6	9.38e-7	2.17e-5
P	409	3.34e-5	4.64e-5	8.72e-5	2.89e-4
B	420	1.78e-4	1.09e-4	4.67e-4	1.79e-4
Frame Type	**Samples**	Stefan (CIF, 30 Hz, QP 39)	Stefan (CIF, 30 Hz, QP 24)	Coastguard (CIF, 30 Hz, QP 39)	Coastguard (CIF, 30 Hz, QP 24)
I	69	4.52e-7	3.3e-6	2.02e-7	1.37e-6
P	409	5.65e-5	3.33e-4	4.32e-5	4.29e-5
B	420	5.09e-4	1.17e-3	9.57e-4	2.40e-4

approach is that we use a parameterization of the decoding tasks that is detailed enough to capture codec specific characteristics but is general enough to be applied to any popular codecs. We deploy a queuing model that enables the system to determine the resource requirements for the set of active applications and also allows for low-complexity estimation of the system state (e.g., queuing and processing delays). Our results show that by explicitly considering the video application's properties in the design of a video decoding system, significant performance gains (in terms of decoded video quality) can be achieved. In particular, our experimental results suggest that the proposed look-ahead risk-aware scheduling policy outperforms a priority-queuing based policy in a highly congested system and, on average, outperforms an earliest-deadline-first policy in an uncongested system. The proposed application and system modeling framework, and the experimental results, provide important insights that can guide the design of multimedia-aware operating systems and hardware as well as future video applications.

There are several interesting extensions to this work. One such extension would be to integrate dynamic-voltage-scaling (DVS) into the proposed framework as a third strategy deployed by the RMCP to change the system dynamics. Through processor frequency adaptation, DVS could be used to adjust the system performance to match the current workload. One way to do this would involve viewing DVS as a tool for adjusting the system's congestion level. Then, using the priority queuing analysis to estimate the expected decoded distortion, the optimal tradeoff between energy consumption and distortion can be explored.

APPENDIX

In this appendix, we describe how we derived the affine transform coefficients and shape parameters shown in Table I to describe the complexity distributions of the I, P, and B frames in the Mobile, Foreman, Stefan, and Coastguard sequences (CIF resolution, 30 Hz, QP 39 and 24). Subsequently, we comment on the validity of the proposed complexity models. Fig. 5 illustrates representative histograms of the measured data for the Mobile sequence with the shifted and scaled Poisson fits superimposed (similar results were obtained for the other sequences in Table I, but are omitted here for brevity). Table VI provides the number of complexity measurement samples and the sample variances for each frame type.

Let $c^{\text{seq}}_{fr(l),QP}$ denote a measured sample of the random variable $C^{\text{seq}}_{fr(l),QP}(n)$. In order to find the Poisson fits shown in Fig. 5, we first normalized $c^{\text{seq}}_{fr(l),QP}$ to obtain $\hat{c}^{\text{seq}}_{fr(l),QP}$ using (5). From (5), $b^{\text{seq}}_{fr(l),QP}$ is determined as the minimum sample complexity and $a^{\text{seq}}_{fr(l),QP}$ is chosen to reflect the sample sets' variance. Given $a^{\text{seq}}_{fr(l),QP}$ and $b^{\text{seq}}_{fr(l),QP}$, we then solved for the shape parameter, $\lambda^{\text{seq}}_{fr(l),QP}$, as the mean of the normalized samples.

In Fig. 5, it is clear that the shifted and scaled Poisson model fits are more accurate for the B frames (right column) than for the P frames (middle column) and I frames (left column). Despite this, we argue that the models are reasonable for the following reasons.

- ***Differential frame complexities:*** Using (11) and the data in Table I, it is clear that the mean I frame complexity is significantly less than the mean P frame complexity which, in turn, is less than the B frame complexity. For example, for the Mobile sequence (CIF resolution, 30 Hz, QP 39), the mean B frame complexity is ~ 1.3 times the mean P frame complexity and ~ 2.2 times the mean I frame complexity. Consequently, accurately modeling the B frame complexity distribution is more important than accurately modeling the P and I frame complexity distributions.
- ***Sample size:*** In Table VI, the sample size for the I frames is significantly smaller than the sample size for the P and B frames. For any GOP structure that is designed to achieve reasonable compression performance, there will always be less I frames than P or B frames. Consequently, the imperfect models for the I frame complexity distributions will have an insignificant impact on the system performance.
- ***Sample variances:*** Observing Table VI, it is clear that the sample variance for the I frame service time measurements [related to the complexity measurements by (8)] is on average an order of magnitude less than the sample variance of the P frame service time samples which, in turn, is nearly an order of magnitude less than the variance of the B frame service time samples. As the sample variance decreases, using a model that captures the mean service time and approximate variance becomes more important than perfectly matching the service time distribution.

In summary, we have found that the affine transformed Poisson distribution is a reasonable model for sequences with a variety of characteristics (i.e., Mobile, Foreman, Stefan and Coastguard) encoded with a practical range of quantization parameter values (i.e., 24 and 39).

REFERENCES

[1] Intel Multi-Core Processors: Leading the Next Digital Revolution [Online]. Available: http://www.intel.com/technology/magazine/computing/multi-core-0905.pdf

[2] M. Hachman, AMD Details Early Quad-Core Plans Extreme Tech, 2005 [Online]. Available: http://www.extremetech.com/article2/0,1558,1826663,00.asp

[3] J. A. Kahle, M. N. Day, H. P. Hofstee, C. R. Johns, T. R. Maeurer, and D. Shippy, "Introduction to the cell multiprocessor," *IBM J. Res. Develop.*, pp. 589–604, Sep. 2005.

[4] M. Horowitz, A. Joch, F. Kossentini, and A. H. Hallapuro, "H.264/AVC baseline profile decoder complexity analysis," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 704–716, Jul. 2003.

[5] V. Lappalainen, A. Hallapuro, and D. Hamalainen, "Complexity of optimized H.26L video decoder implementation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 717–725, Jul. 2003.

[6] W. Yuan, K. Nahrstedt, S. V. Adve, D. L. Jones, and R. H. Kravets, "GRACE-1: Cross-layer adaptation for multimedia quality and battery energy," *IEEE Trans. Mobile Comput.*, vol. 5, no. 7, pp. 799–815, Jul. 2006.

[7] W. Yuan and K. Nahrstedt, "Energy-efficient soft real-time CPU scheduling for mobile multimedia systems," in *Proc. 19th ACM Symp. Operating Systems Principles*, 2003, pp. 149–163.

[8] T. Stockhammer, M. M. Hannuksela, and T. Wiegan, "H.264/AVC in wireless environments," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 657–673, Jul. 2003.

[9] E. B. van der Tol, E. G. Jaspers, and R. H. Gelderblom, "Mapping of H.264 decoding on a multiprocessor architecture," in *Proc. SPIE Conf. Image and Video Communications and Processing*, Jan. 2003.

[10] S. Dutta, R. Jensen, and A. Rieckman, "Viper: A multiprocessor SOC for advanced set-top box and digital TV systems," *IEEE Design Test Comput.*, vol. 18, no. 5, pp. 21–31, Sep.–Oct. 2001.

[11] D. Bertsekas and R. Gallager, *Data Networks*. Upper Saddle River, NJ: Prentice-Hall, 1987.

[12] M. van der Schaar and Y. Andreopoulos, "Rate-distortion-complexity modeling for network and receiver aware adaptation," *IEEE Trans. Multimedia*, vol. 7, no. 3, pp. 471–479, Jun. 2005.

[13] P. A. Chou and Z. Miao, "Rate-distortion optimized streaming of packetized media," *IEEE Trans. Multimedia*, vol. 8, no. 2, pp. 390–404, Apr. 2006.

[14] M. van der Schaar and D. S. Turaga, "Cross-layer packetization and retransmission strategies for delay-sensitive wireless multimedia transmission," *IEEE Trans. Multimedia*, vol. 9, no. 1, pp. 185–197, Jan. 2007.

[15] D. Jurca and P. Frossard, "Packet selection and scheduling for multipath video streaming," *IEEE Trans. Multimedia*, vol. 9, no. 2, pp. 629–641, Apr. 2007.

[16] Y. Wang, M. van der Schaar, S.-F. Chang, and A. C. Loui, "Classification-based multidimensional adaptation prediction for scalable video coding using subjective quality evaluation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 15, no. 10, pp. 1270–1279, Oct. 2005.

[17] P. Pillai and K. Shin, "Real-time dynamic voltage scaling for low-power embedded operating systems," *SIGOPS Oper. Syst. Rev.*, vol. 35, no. 5, pp. 89–102, Dec. 2001.

[18] T. Jiang, C. K. Tham, and C. C. Ko, "An approximation for waiting time tail probabilities in multiclass systems," *IEEE Commun. Lett.*, vol. 5, no. 4, pp. 175–177, Apr. 2001.

[19] J. Abate, G. L. Choudhury, and W. Whitt, "Exponential approximations for tail probabilities in queues I: waiting times," *Oper. Res.*, vol. 43, no. 5, pp. 885–901, 1995.

[20] A. Burns, "Scheduling hard real-time systems: A review," *Software Eng. J.*, vol. 6, no. 3, pp. 116–128, May 1991.

[21] D. Gross and C. M. Harris, *Fundamentals of Queuing Theory*, 3rd ed. New York: Wiley, 1998.

[22] L. Kleinrock, *Queuing Systems Volume I: Theory*. New York: Wiley-Interscience, 1975.

Nicholas H. Mastronarde (S'07) received the M.S. degree in electrical engineering from the University of California at Davis in 2006. He is currently pursuing the Ph.D. degree in the Electrical Engineering Department at the University of California at Los Angeles.

His research interests are in multimedia systems, multimedia complexity modeling, and multimedia system software design

Mihaela van der Schaar (SM'04) is currently an Associate Professor in the Electrical Engineering Department at the University of California, Los Angeles (UCLA). Since 1999, she was an active participant to the ISO MPEG standard to which she made more than 50 contributions and for which she received three ISO recognition awards.

Dr. van der Schaar received the NSF CAREER Award in 2004, IBM Faculty Award in 2005, Okawa Foundation Award in 2006, Best IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY Paper Award in 2005, and Most Cited Paper Award from EURASIP Journal Signal Processing: Image Communication between the years 2004–2006. She holds 28 granted U.S. patents. She is currently an associate editor of IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, IEEE SIGNAL PROCESSING LETTERS and *IEEE Signal Processing e-Newsletter*. She is also the editor (with Phil Chou) of the book *Multimedia over IP and Wireless Networks: Compression, Networking, and Systems*.