Joint Scheduling and Resource Allocation for Multiple Video Decoding Tasks

Brian Foo and Mihaela van der Schaar

Dept. of Electrical Engineering (EE), UCLA
66-147E Engineering IV Building, 420 Westwood Plaza,
Los Angeles, CA, 90095-1594, USA

ABSTRACT

In this paper we propose a joint resource allocation and scheduling algorithm for video decoding on a resource-constrained system. By decomposing a multimedia task into decoding jobs using quality-driven priority classes, we demonstrate using queuing theoretic analysis that significant power savings can be achieved under small video quality degradation without requiring the encoder to adapt its transmitted bitstream. Based on this scheduling algorithm, we propose an algorithm for maximizing the sum of video qualities in a multiple task environment, while minimizing system energy consumption, without requiring tasks to reveal information about their performances to the system or to other potentially exploitative applications. Importantly, we offer a method to optimize the performance of multiple video decoding tasks on an energy-constrained system, while protecting private information about the system and the applications.

Keywords: *information privacy, priority scheduling, queuing theory, decentralized algorithms, complexity-scalable multimedia applications*

1. INTRODUCTION

With the advent of web TV, YouTube, peer-to-peer multimedia streaming, online gaming, etc., multiple autonomous multimedia processing, compression, transcoding, and streaming tasks need to be executed simultaneously on the same system. Multimedia applications are scalable in the sense that different video qualities can be achieved under different amounts of allocated network rate and processing resources. Based on the stochastic analysis of video frames at the encoder, or statistical training at the decoder, the tradeoff between the average video quality and video decoding complexity of a sequence can be accurately modeled [1] [2]. These curves enable the system to smartly divide limited computational resources among multiple video decoding tasks in a way that maximizes an overall application-dependent system objective, such as the sum of video qualities [3] [4].

Numerous features exist in current video coding standards that enable both proactive and online workload adaptation and estimation for video sequences based on feedback from the decoding device [5] [6]. In general, a continuous exchange of information between the encoding and decoding devices is needed to ensure that the workload is feasible at any given time instance. To overcome this limitation, in our previous work [2], we proposed a "self-contained" priority-scheduling algorithm for the decoder that can *gracefully adapt* to time-varying video characteristics, even when the received workload is infeasible. A major advantage of this scheduling algorithm is that the decoder does not need to continuously update encoding devices about its current energy consumption (or CPU utilization) and force a change in transmission policies at the encoder end. Essentially, an efficient quality versus complexity tradeoff curve can be obtained by the decoder alone based on system constraints, and hence the (networked or remote) encoder does not require any information regarding the system's energy or computational

resource constraints. This is an important requirement for systems that may want to protect private information about their resources from multimedia streaming services.

In this paper, we extend the idea of private information to a resource allocation problem involving multiple decoding tasks running on the same system. Based on the priority-scheduling scheme in [2], we propose an informationally-decentralized resource allocation scheme that provides an optimal, application-dependent, centralized resource allocation solution. While the solution involves a central resource manager, the resource manager exists only to exchange and update decoding tasks about system *resource* usage and congestion using very low complexity operations. Importantly, each decoder's algorithms and their resulting energy-quality tradeoff curves need not be communicated to the resource manager, or to any of the other simultaneously running applications. This type of information privacy is ideal for shared systems, where competing companies may have incentives to create products that can exploit information about the coding algorithms and performance curves of other applications in order to gain an unfair performance advantage.

The organization of our paper is as follows: Section 3 introduces the priority scheduling algorithm, and the queuing-theoretic analysis necessary to determine the energy-quality tradeoff curve for the application. Section 4 provides both the centralized and decentralized resource allocation schemes based on the energy model constructed by the underlying coding and scheduling scheme. Section 5 provides simulations that compare the performances of algorithms and their convergence times, and Section 6 concludes our paper.

2. OVERVIEW OF THE PROPOSED ALGORITHM

Applications such as teleconferencing and mobile surveillance require the decoding of multiple multimedia tasks received from a network of devices and/or cameras (See Figure 1). Such devices require efficient resource allocation and scheduling solutions to reduce energy and CPU cycle consumption. While various solutions have been proposed to jointly schedule multiple decoding tasks on the same system, such solutions are often limited in that they do not consider explicitly (and analytically) the effect of such scheduling algorithms on the overall quality of multimedia applications. Moreover, there is little discussion on a resource allocation solution that maximizes some overall, application-dependent system utility.

In this paper, we propose a joint resource allocation and scheduling scheme following the procedure given in Table 1. By collecting information from training sequences or from prior observed workloads, the application builds models for the complexity of each video frame. (See our prior work for an approach for modeling complexity [2].) Using queuing theoretic analysis and decomposing the frames (or jobs) into different priority classes, the application can build a model of the average video quality as a function of available resources. For example, the application can count the probability that jobs in a priority class miss their deadlines, and multiply that probability to the average quality contribution for that particular class to obtain the quality degradation per class. By varying the quality levels (or processor power levels), each application obtains different energy-quality points to construct an energy-quality curve. The system then provides a resource allocation solution to maximize the sum of video qualities for multiple applications sharing the system by using each application’s energy-quality curve. Finally, the system schedules each application based on the amount of resources provided under the optimal allocation solution. Note that in this work, in order to focus on resource allocation for video decoding, we consider only the energy consumed by the video decoding algorithms, and not the energy required by the encoding devices to encode or transmit the video bitstream. In other words, we assume that the received bitstream from the network remains unaffected and hence the energy consumption for communications is fixed. However, scalable transmission algorithms can also be incorporated into this resource allocation scheme by using similar models. The individual steps of the procedure in Table 1 is explained in more detail in the following two sections.

Figure 1 Examples of energy-limited mobile surveillance (left) and teleconferencing (right).

Table 1 Summary of the joint resource allocation and scheduling scheme for multiple video decoding.

1.	App. collects workload information from training sequences, or by updating based on prior workloads.
2.	App. builds complexity models for each priority class of video frames.
3.	App. estimates its video quality based on queuing delay analysis and resource constraints.
4.	App. Builds an energy-quality tradeoff curve.
5.	System maximizes sum of video qualities for all apps., based on their energy-quality curves and energy availability.
6.	System schedules the applications's jobs.

3. DERIVING ENERGY-QUALITY CURVES BASED ON SMART SCHEDULING

3.1. Priority Scheduling of Video Decoding Tasks to Minimize Energy Consumption

In most state-of-the-art video encoding schemes, video frames are jointly coded using groups of pictures (GOP) or frames. For example, the MPEG standard uses I, B, and P frames, while H.264 uses dynamic GOP structures. The dependencies between frames in a GOP do not only determine the order in which frames must be decoded (i.e. the priority), but each type of encoded frame further down the hierarchy also contributes a decreasing amount to the overall video quality. In this section, rather than servicing jobs based on earliest deadline first (EDF), we propose servicing jobs based on quality-based priority levels, such that a decent quality level can be achieved even if not all frames can be decoded. Figure 2 provides an example GOP structure for 3 temporal-level motion-compensation temporal filtering (MCTF). In Figure 3, we provide an example of a priority-based job decomposition of the MCTF GOP structure in Figure 2.

In [2], it was shown that entropy decoding (ED) complexity from the bitstream can be modeled as incoming *groups of cycles* (GOCs) that arrive according to a memoryless (Poisson) process. Each GOC can be interpreted as data that is decoded from the bitstream and is used in further processing steps, such as inverse transform, motion compensation, and fractional pixel interpolation. Based on the decomposition of jobs into arriving ED GOCs, we can model the priority scheduling decoding process as a non-preemptive $M/G/1$ queuing system with GOCs as packets, where the service time distribution is given by the time required to perform all required operations on the GOC. Priority scheduling ensures that even if not all jobs can be processed before the display deadline, the higher priority jobs will be processed first, such that they are more likely to satisfy their deadline constraints. Effectively,

this enables the system to gracefully adapt the video quality under different resource constraints, or essentially, different frequencies provided that the processor enables voltage scaling. Note that the priority scheduling algorithm shown in Figure 3 may cause a low priority frame with an earlier deadline to be missed at the cost of decoding a high priority frame with later deadline. Hence, there are always cases where priority-scheduling based on quality contribution is suboptimal. However, it is important to note that without this scheme, almost all frames will be dropped by an overloaded system, since the delay will be huge. The priority scheme, on the other hand, enables important frames to be decoded regardless of the total workload, and hence scales with different system resource constraints.

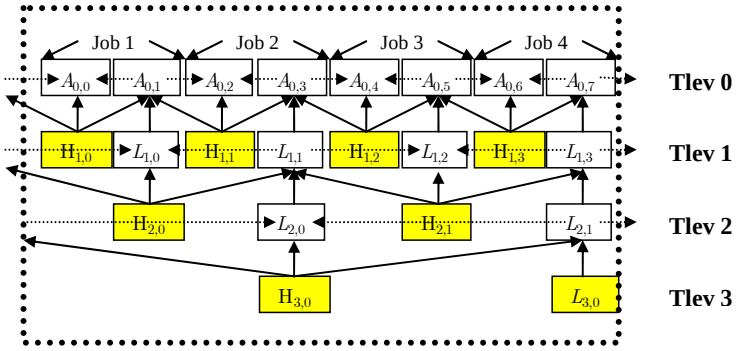

Figure 2: A deadline-based job representation for an MCTF-encoded video GOP. The top row indicates the displayed video frames. The frames received from the bitstream are the H-frames, and the L-frame at highest temporal level. Arrows indicate the dependencies in video decoding.

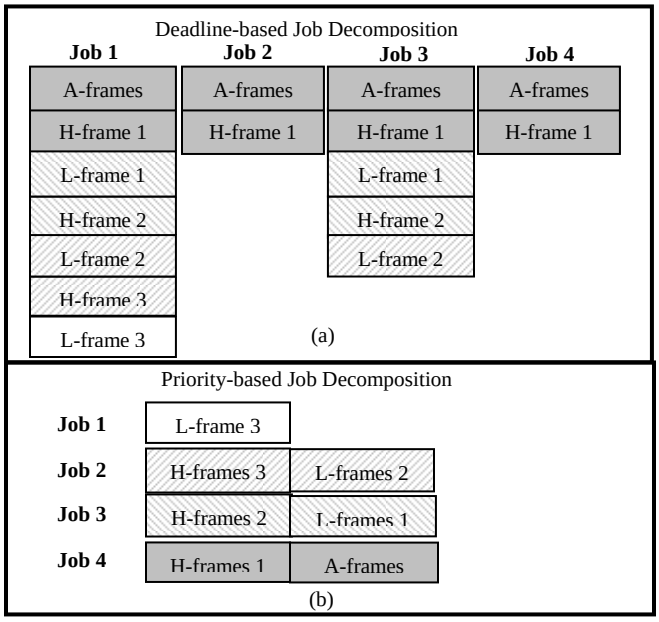

Figure 3: (a) Deadline-based job decomposition and (b) Priority-based job decomposition for 3 temporal level MCTF. The temporal level is indicated by the number beside the frame type.

To obtain energy-quality tradeoff curves, we reverse the problem such that the objective is to find a minimum operating power level that can meet a required average video quality level, i.e.:

$$\begin{aligned} &\min P_k \\ &s.t. Q_k \geq Q_{\text{avg}} \end{aligned} \tag{1}$$

where:

$$Q_k = \sum_{i=1}^{I} \frac{\lambda_i}{\lambda} \Delta_i \Pr\{D_{i,k} \le t_i\} \tag{2}$$

is the average quality of the decoded sequence at power level P_k. Here, λ_i is the intensity of arrivals of GOCs with priority i, and λ is the intensity of arrivals of all GOCs. Hence, λ_i / λ is the fraction of GOCs with priority i received from the bitstream. Δ_i denotes the quality slope parameter for priority i frames (i.e. the average quality contributed to video by a priority i frame.), which can be estimated from the variance of coefficients in the subbands of the frames. (See [7] for some details regarding the quality slope.). The computation of $\Pr\{D_{i,k} \le t_i\}$, which is the probability that the delay of a job of priority class i under processor mode k, $D_{i,k}$, will make the delay deadline t_i, will be discussed in the following subsection.

3.2. Delay Analysis for Non-preemptive M/G/1 Priority Queuing

In this section, we provide an analytical approximation for the delay associated with each priority class of jobs. This analysis will enable us to determine explicitly the quality Q_k using (2), and thus solve the optimization problem in (1). Let $D_{i,k}$ be the delay of processing a GOC of class i, and define $\Pr\{D_{i,k} > T_i\}$ to be the probability that a GOC arriving at time t can not be processed before deadline $t + T_i$. Note that in reality, all GOCs for the same job (e.g. video frame) have the same hard deadline regardless of their arrival times t, so the delay bound T_i would not be fixed for every GOC of a priority class i. However, considering that GOCs need to be processed in FIFO order to complete the job, the deadlines for the first GOCs in the job may be set earlier to accommodate the processing time delay induced on later GOCs. For the purpose of analysis, we approximate the delays tolerated by all GOCs for a class i to be approximately equal.

In order to determine the probability of violating the delay deadline for a non-preemptive priority queuing system, we first define the load on the system induced by priority class i with service time $S_{i,k}$ as:

$$\rho_{i,k} = \lambda_i E[S_{i,k}] \tag{3}$$

Let $\sigma_{i,k} = \sum_{j=1}^{i} \rho_{j,k}$ be the total load of traffic coming from priority classes 1 to i, and let $\mu_{i,k}$ be the average service rate for a class i job in processor operating mode k. The average waiting time in the queue for priority class i GOCs can then be expressed as [8]:

$$\mathrm{E}[W_{i,k}] = \frac{\sum_{j=1}^{I} \frac{\rho_{j,k}}{\mu_{j,k}}}{2(1-\sigma_{i-1,k})(1-\sigma_{i,k})} \tag{4}$$

From the average waiting time, we can obtain an approximation for the probability that the waiting time exceeds some time t. We use the waiting time tail approximation to estimate the tail of the delay [9]:

$$\Pr\{D_{i,k} > T_i\} = \Pr\{W_{i,k} + S_{i,k} > T_i\} \approx \rho_k \exp\left(-\frac{\rho_k T_i}{\mathrm{E}[W_{i,k}] + E[S_{i,k}]}\right) \tag{5}$$

We summarize the algorithm used to compute the energy-quality tradeoff curves in Table 2.

4. RESOURCE ALLOCATION TO MULTIPLE TASKS

4.1. Centralized Resource Allocation Problem for Multiple Tasks

By setting Q_{avg} to different thresholds in (1), it is possible to obtain a set of energy-quality tradeoff curves for individual tasks based on the proposed priority scheduling algorithm. In this section, we consider a solution that enables the system to allocate CPU time to multiple tasks, based on the operating points of each task's energy-quality tradeoff curves. This solution maximizes the *social welfare*, defined by the sum of video qualities, while

minimizing the overall system energy consumption. Denote the quality level for task i, as a function of its allocated resources (e.g. cycles) per unit time, x_i, by $Q_i(x_i)$. Also, denote the minimum energy consumed per unit time (i.e. the average power), for a total rate of resource consumption x, by $E^*(x)$. We define the following objective function for maximizing system performance:

$$\max_{x_i} \sum_{i=1}^{I} Q_i(x_i) - vE^*\left(\sum_{i=1}^{I} x_i\right). \quad \text{s.t. } x_i \geq 0 \tag{6}$$

where $v > 0$ is a weighting factor that determines the importance of saving energy, which ultimately determines the resource allocation for each multimedia application that maximizes the total quality of tasks under different energy consumption levels. For demonstration purposes, we chose to use the social welfare function. However, we can similarly maximize an application-dependent fairness metric based on the Nash criterion [10]. In this case, we want to maximize the product of the video qualities. This is equivalent to maximizing the sum of logs of the utilities, i.e.:

$$\max_{x_i} \ln\left(\prod_{i=1}^{I} Q_i(x_i)\right) - vE^*\left(\sum_{i=1}^{I} x_i\right) = \max_{x_i} \sum_{i=1}^{I} \ln Q_i(x_i) - vE^*\left(\sum_{i=1}^{I} x_i\right), \quad \text{s.t. } x_i \geq 0 \tag{7}$$

which can be solved using the same algorithms proposed in this paper.

In [1], it was shown that complexity-quality curves are concave. Moreover, from [12], the complexity-energy curve is convex. Hence, the optimization problem in (6) (and (7)) is convex, and convex optimization approaches (e.g. SQP [13]) can be used to find the optimal resource allocation scheme. However, the centralized algorithm requires a resource management program (RMP) to collect information regarding the energy-quality tradeoff points for each application, which we have assumed to be private information that is preferably not shared with the system. Hence, an informationally-decentralized approach for resource allocation is needed.

Table 2 Algorithm for energy-quality tradeoff curve computation

For each job class
  Calculate the average quality contribution Δ_i and the complexity distribution.
  Determine service rate for each job class under power level P_k (e.g. available CPU cycles)
End For
For each power level k
  Estimate the probability of dropping frames for each class using (4) and (5).
End For
For different quality levels
  Solve (1) to find the minimum power level.
  Determine the energy consumed.
End For

4.2. Decentralized Approach for Resource Allocation for Multiple Tasks

We propose an iterative, informationally-decentralized algorithm that optimizes (6) without requiring each multimedia task to submit information about its scheduling algorithm or its resulting energy-quality tradeoff points. While a central RMP is still required to collect some information, each task submits only its resource demand x_i to the RMP. Based on the energy availability (supply), the RMP calculates a tax function that penalizes each task according to the excess energy it adds to the system:

$$t_i(x_i) = vE^*(x_i + d_i), \tag{8}$$

where:

$$d_i = \sum_{l \neq i}(x_l') + \frac{1}{\gamma}\sum_{l \neq i}(x_l - x_l'). \quad (9)$$

Note that the first term in (9) is the resource demand from all other tasks. However, the second term is added in to "slow down" the change in resource demands during each iteration by a factor of $1/\gamma$. We denote d_i as task i's *perceived* resource demand from all other users. The parameter γ can be adjusted after each iteration. After receiving the tax function, task i then performs an individual optimization:

$$x_i = \arg\max_{x \geq 0}\{Q_i(x) - t_i(x)\}, \quad (10)$$

and submits its new resource demand x_i back to the RMP. Note that (10) is a simple optimization of a single variable and can discover the optimal point quickly using Newton search methods. The tax function is also a very low complexity calculation to be performed by the RMP. The process of making demands and updating the tax function iterates until convergence, or until the program terminates. From here on, we denote the tax function in (8) as the excess-energy minimizing (EEM) tax function. The algorithm is provided in Table 3.

An important property to consider is whether the iterative algorithm converges to the optimal solution of the centralized problem in (6), and how quickly it converges (e.g. adaptation rate to dynamics). We omit a rigorous proof of convergence here, as similar proofs can be found in [15], but we note that convergence is guaranteed as long as the sum of the sequence generated by $1/\gamma$ with each iteration is infinite (e.g. $1/n$) [15]. Below, we provide a proof that (10) generates an optimal solution for (6) upon convergence. The proof is based on the existence and uniqueness of a point that satisfies the Karush-Kuhn-Tucker conditions for (6), and the fact that the same KKT conditions uniquely and simultaneously satisfy the KKT conditions of (10) when resource demands stop changing.

Proposition 1: *For strictly concave quality functions and convex energy functions, the EEM tax function generates an optimal solution to the centralized problem in* (6) *upon convergence.*

Proof: See Appendix A. ■

Another important property to consider is whether the algorithm adapts quickly to dynamics. For example, in video, the utility function is highly time-varying depending on the content of the video. If a near optimal resource allocation can be obtained using very few iterations of the low complexity, EEM-based taxing algorithm, this may be preferred over repeated solving the centralized problem of n variables each time the utility function changes.

Table 3 Iterative, informationally-decentralized EEM-based algorithm

RMP: Initialize a resource allocation scheme (e.g. fair sharing $x_i = C/n$, n is the number of tasks).
Loop until termination of program {
  RMP: submit a tax function based on (8) to each task.
  For each task
    Task: calculate new x_i based on (10).
  End For
}

4.3. Regarding Multiplexing of Multiple Multimedia Tasks

While we have proposed a scheduling scheme for a single task, and resource allocation for multiple tasks, we have not considered the possibility of jointly scheduling jobs between different applications. Unfortunately, optimally scheduling/multiplexing jobs between different applications on a single processor is very difficult because

the underlying coding algorithms and parameters for each application may be different, and/or may vary over time. More importantly, the autonomous applications may regard their coding and scheduling algorithms as private information, and hence their time-varying resource requirements are not shared with the RMP or with other tasks. Based on these conditions, it is necessary to break this problem (as we have done) into an informationally-decentralized resource allocation problem between tasks, and a scheduling problem for each task.

A possible limitation to our joint resource allocation and scheduling approach is that the delay estimates for each application, and hence the energy-quality tradeoff curves, may not be precise depending on the scheduling algorithm between tasks. In particular, we have provided delay estimates for each task running in isolation, but not the delay as a result of multiple tasks sharing a processor. Nevertheless, the delay estimate will be accurate if generalized processor sharing (GPS) is used. In GPS, each task is processed as though it were on an independent processor with a speed equal to the fraction of the resources it is using from the real processor. While GPS is an ideal scheduling algorithm, various other algorithms, such as weighted round-robin scheduling, have been used to approximate GPS [14]. Since the time slices for these algorithms are typically on the order of a few milliseconds, such small discrepancies in the delay estimates will not affect much the performance of our algorithms.

5. SIMULATIONS AND RESULTS

To illustrate our joint scheduling and resource management algorithm, we used the video coder from [11] in our simulations to encode various sequences at different bit rates, and we gathered the number of cycles for each decoding job on a Pentium IV processor. We note that while we have chosen a specific coder, our methodologies can just as easily apply to other complexity scalable video coding algorithms. In order to verify that our algorithms are suitable for DVS-enabled PEs, we used the StrongARM processor profile given in [16]. The StrongARM processor enables 11 different voltage levels, thereby enabling us to apply our scheduling and resource allocations to a finer granularity.

5.1. Energy Scalability of the Priority-Scheduling Algorithm

To give a reference point for the performance of our scheduling algorithm, we compared against the look-ahead earliest deadline first (laEDF) DVS algorithm proposed in [17]. We decided to use laEDF since, unlike many other real-time scheduling algorithms (e.g. see [18] [19]), laEDF provides an aggressive DVS energy minimization approach that *jointly* considers the worst-case execution times of different types of jobs in the processing queue. To determine the benefit derived from our queuing model, we also provided an EDF algorithm based on similar queuing analysis (QEDF) as in Section 3.2, and operated at the minimum dynamic power levels such that almost all frames were able to meet their display deadlines (See [2] for more details on the QEDF algorithm.). The QEDF outperforms the QDPS algorithm when the energy is sufficient, since video frames are never decoded out of order. Comparing laEDF with QEDF side by side on the top row of Table 4 shows that the energy savings using an accurate queuing model is approximately 11-13%.

Secondly, we determine the benefit of quality-driven priority scheduling (QDPS) algorithm by comparing the performance of laEDF with QDPS in Table 4 when the energy level is set lower. Note from Table 4 that EDF-based scheduling policies do not scale with low power, since setting the power too low will inevitably cause the delay to be intolerable for all frames due to the large number of jobs queued in the buffer. Effectively, under a certain energy level, the video frames will all miss their display deadlines. On the other hand, our priority scheduling algorithm ensures that high priority transform frames will be decoded even if not all frames can be decoded. Hence, we decode only the most important frames under low power and thus achieve high scalability in terms of quality and energy tradeoffs. For example, the QPDS incurs only a loss in quality of less than 1.5 dB when the power is scaled down to 10% the original power. Likewise, when the energy is scaled down to 25% the original power, the quality

degradation is less than 1.0 dB. On the other hand, laEDF has an average PSNR degradation of almost 9 dB for when the power is scaled to 75%, and drops nearly all frames when power is decreased to around 25%! Finally, note that for certain intermediate power levels (e.g. 2.15E), the QPDS algorithm drops more frames than laEDF for the Stefan sequence due to out-of-order scheduling. However, QPDS still achieves significantly higher PSNR, since it decodes the most important layers of the GOP first, while laEDF will fully decode some video frames while completely dropping other frames due to delay deadline violation.

Table 4 Comparisons of the energy consumption of laEDF DVS with QPDS for various energy consumption levels using the Coastguard and Stefan sequences decoded a bit rate 768kbps. The top row compares the energy consumption of laEDF and QEDF for decoding all frames.

	Energy Consumed		**PSNR**		**Frame Rate**	
Algorithms	***Coastguard***	***Stefan***	***Coastguard***	***Stefan***	***Coastguard***	***Stefan***
laEDF - QEDF	2.90E - 2.63E	2.72E - 2.41E	33.24 - 33.24	27.35 - 27.35	30.00 - 30.00	30.00 - 30.00
laEDF - QDPS	2.15E (74%)	2.15E (79%)	26.00 - 32.98	**23.44 - 27.01**	23.46 - 26.48	**25.72 - 23.67**
laEDF - QDPS	1.26E (43%)	1.25E (46%)	15.12 - 32.51	14.05 - 26.70	13.64 - 20.04	15.58 - 18.05
laEDF - QDPS	0.65E (22%)	0.65E (28%)	**0.00** - 32.23	**0.00** - 26.48	0.00 - 16.17	0.00 - 15.23
laEDF - QDPS	0.28E (9.6%)	0.29E (11%)	**0.00** - 32.05	**0.00** - 25.94	0.00 - 14.53	0.00 - 10.08
laEDF - QDPS	0.09E (3.1%)	0.09E (3.3%)	**0.00** - 30.68	**0.00** - 25.55	0.00 - 8.09	0.00 - 7.27
laEDF - QDPS	0.02E (0.7%)	0.03E (1.1%)	**0.00** - 29.44	**0.00** - 24.62	0.00 - 5.04	0.00 - 3.63

5.2. Quality-Power Levels based on the Tradeoff Parameter v

In the following simulations, we considered a system where 2 *Foreman*, 2 *Coastguard*, and 1 *Mobile* sequences are being decoded using the QPDS algorithm. Note that only one curve for *Foreman* and *Coastguard* are plotted, since the resource allocation is identical for the same sequences. In Figure 4, the optimal quality-energy solution is compared against the application-agnostic fair-share resource allocation scheme (i.e. all tasks are allocated equal shares of CPU time) for various values of v. Note that the EEM allocates more resources to the *Mobile* sequence since its performance increases drastically with increased resources. On the other hand, Coastguard, which benefits less from increased resources, receives less total resources and therefore has lower quality. Overall, the sum of PSNRs for all tasks, shown in Figure 4b, is 5-10 dB higher for the EEM than for fair share resource allocation at low-power regions, which translates to a significant 1-2 dB PSNR higher average video quality per task.

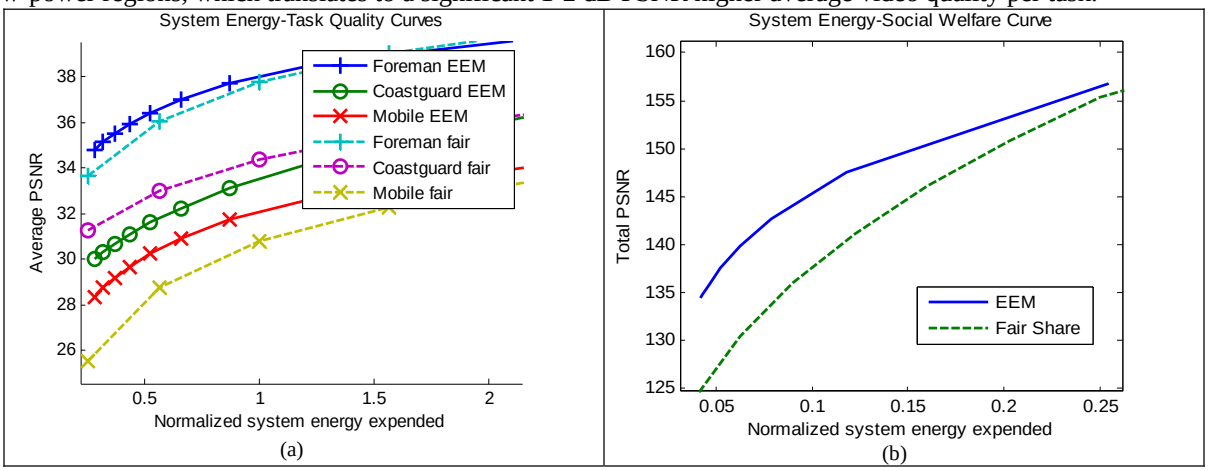

Figure 4 (a) Plots of the quality of each decoded video based on the total energy expended by the system, compared between our resource allocation scheme versus fair-share allocation. (b) The corresponding plot of the total utility of all tasks.

In Figure 5, we plotted the resulting video qualities and resource allocations as a function of the iterations in the EEM algorithm. Each iteration corresponds to an exchange of supply and demand between each task and the CPU. The step size parameter for the EEM tax function was set to $\gamma = 1$, and the plots are given for $v = 2 \times 10^{-3} \mathrm{PSNR/Joule}$. Note that the algorithm converges to the centralized solution in only about 5 iterations and is therefore ideal for updating resource allocation in dynamic environments.

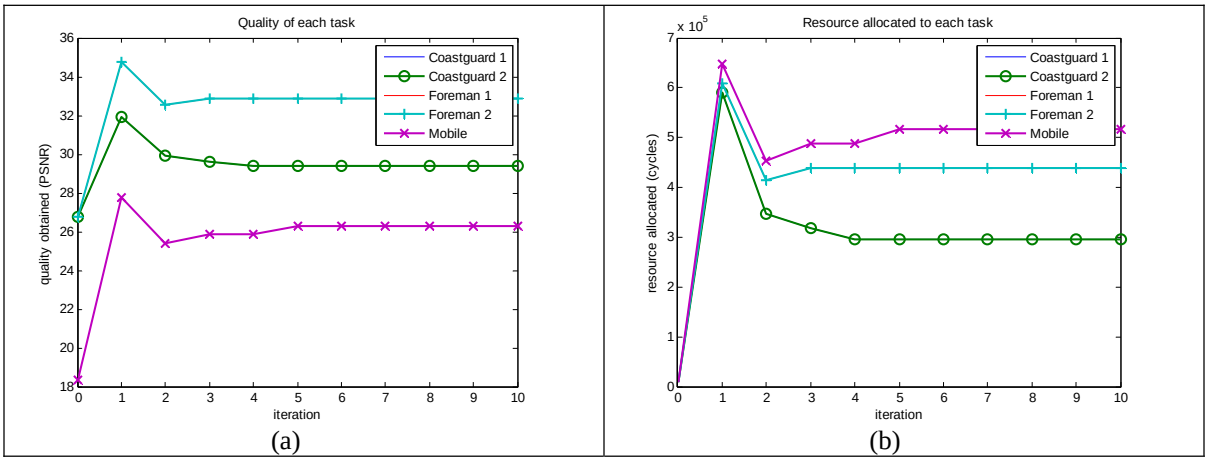

Figure 5 (a-b) The convergence of quality and computational resource allocation for the decentralized algorithm (EEM).

6. CONCLUSION

In this paper, we presented a joint scheduling and resource allocation solution for multiple multimedia tasks sharing the same resource-constrained system. We demonstrated that our proposed algorithm can obtain highly energy-scalable solutions for multiple video decoding applications while protecting the private information of these tasks from the system and from encoding devices. We demonstrated that this joint approach can greatly improve the performance of all video decoding tasks sharing the system under stringent delay and energy constraints, and outperforms existing scheduling and resource allocation solutions. Avenues for future work include constructing multiplexing algorithms for video decoding tasks, and performing joint scheduling and resource allocation for multiprocessor systems. Also, the current work provides insights into the design of token-based or monetary-based penalties to prevent tasks with private information from cheating the system. The robustness of the system toward strategically-behaving tasks can be further investigated in future work.

REFERENCES

[1] B. Foo, Y. Andreopoulos, and M. van der Schaar, “Analytical Rate-Distortion-Complexity Modeling of Wavelet-based Video Coders,” IEEE Trans. Signal Process., to appear.

[2] B. Foo, M. van der Schaar. “A Queuing Theoretic Approach to Processor Power Adaptation for Video Decoding Systems,” *IEEE Trans. Signal Processing*, to appear.

[3] Ghosh, S., Rajkumar, R. R., Hansen, J., and Lehoczky, J. (2003). “Scalable resource allocation for multi-processor QoS optimization,” In *23rd IEEE International Conference on Distributed Computing Systems* (ICDCS 2003).

[4] L. Capra, W. Emmerich, C. Mascolo, “CARISMA: Context-aware reflective middleware system for mobile applications,” *IEEE Trans. on Software Engineering*, Vol. 29, No. 10, Oct. 2003.

[5] 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, June 2005.

[6] J. Valentim, P. Nunes, and F. Pereira, “Evaluating MPEG-4 video decoding complexity for an alternative video verifier complexity model,” IEEE *Trans. Circuits Syst. Video Technol.*, vol. 12, no. 11, pp. 1034–1044,

Nov. 2002.
[7] A. Ortega, K. Ramchandran. “Rate-distortion Methods for Image and Video Compression,” *IEEE Signal Processing Mag.*, vol. 15, issue 6, pp. 23-50, Nov, 1998.
[8] D. Gross and C. Harris, *Fundamentals of Queueing Theory*, New York: Wiley-Interscience, 1997.
[9] J. Abate, G. L. Choudhury, and W. Whitt. “Exponential approximations for tail probabilities in queues I: Waiting times”, *Operations Research*, vol. 43, no. 5, pp 885-901, 1995.
[10] J. Nash, “The Bargaining Problem,” Econometrica, 1950.
[11] B. Foo, Y. Andreopoulos, M. van der Schaar. “Analytical Complexity modeling of Wavelet-based Video Coders,” *ICASSP* 2007.
[12] T. Ishihara, H. Yasuura. “Voltage Scheduling Problem for Dynamically Variable Voltage Processors,” in *Proc. ACM ISLPED*, 1998, pp. 197-202.
[13] D. P. Bertsekas, “Nonlinear Programming,” Belmont, MA: Athena Scientific, 2nd Edition, 1999.
[14] L. Kleinrock, “Time-shared Systems: a theoretical treatment,” *JACM*, Vol. 14, Issue 2, Apr. 1967.
[15] N. A. Lynch, *Distributed Algorithms.* San Mateo, CA: Morgan Kaufmann, pp. 51-80, 1996.
[16] A. Sinha and A. P. Chandrakasan, “Jouletrack: A web based tool for software energy profiling,” in *Proc. IEEE/ACM DAC*, pp. 220–225, 2001.
[17] P. Pillai, K. Shin. “Real-Time Dynamic Voltage Scaling for Low-Power Embedded Operating Systems.” Proceedings of the eighteenth ACM symposium on Operating Systems, 2001.
[18] Vijay Raghunathan, Papleologos Spanos, Mani B. Srivastava, "Adaptive Power-Fidelity in Energy-Aware Wireless Embedded Systems," rtss, p. 106, *22nd IEEE Real-Time Systems Symposium (RTSS'01),* 2001.
[19] Q. Wu, P. Juang , M. Martonosi , D. Clark, “Formal online methods for voltage/frequency control in multiple clock domain microprocessors,” *Proceedings of the 11th international conference on Architectural support for programming languages and operating systems*, Oct. 2004.

APPENDIX A: PROOF OF PROPOSITION 1

Proof: To show that the decentralized task objective functions are at equilibrium at the optimal point of the system-wide objective function in (6), and that the equilibrium point is unique, we use the Karush-Kuhn-Tucker (KKT) conditions for optimality. Given that the system-wide objective function in (6) is concave, the sufficient KKT conditions for optimality in (6) are that there exist $\mu = (\mu_1, \mu_2, ..., \mu_I) \geq 0$, such that:

$$\begin{aligned} &\nabla\left(\sum_{i=1}^{I} Q_i(x_i) - E^*\left(\sum_{i=1}^{I} x_i\right)\right) - \mu = 0 \\ &\mu_i r^{(i)} = 0, \quad i = 1,...,I \end{aligned} . \tag{11}$$

The KKT conditions for the task level optimization using the EEM tax function (8) are:

$$\begin{aligned} &\frac{d}{dx_i} Q_i(x_i) - \frac{d}{dx_i} E^*\left(\sum_{i=1}^{I} (x_i + d_i)\right) - \mu_i = 0 \\ &\mu_i x_i = 0 \end{aligned} . \tag{12}$$

For all users to be at an equilibrium point for the decentralized algorithm, the change in demand for each user should be 0, and thus based on (9), the excess demand $d_i = \sum_{l \neq i} x_l$. Hence at equilibrium, we have the following condition:

$$\begin{aligned} \frac{d}{dx_i} Q_i(x_i) - \frac{d}{dx_i} E^*(x_i + d_i) - \mu_i &= \frac{\partial}{\partial x_i} Q_i(x_i) - \frac{\partial}{\partial x_i} E^*\left(x_i + \sum_{l \neq i} x_l\right) - \mu_i \\ &= \nabla_i \left(\sum_{i=1}^{I} Q_i(x_i) - E^*\left(\sum_{i=1}^{I} x_i\right)\right) - \mu_i \end{aligned} . \tag{13}$$

Since the intersection of all KKT conditions for task objective functions at equilibrium (13) is the same condition for optimality of the global system objective (11), convergence of the decentralized algorithm (EEM) guarantees an optimal solution to the global objective function.