Informationally Decentralized System Resource Management for Multiple Multimedia Tasks

Brian Foo and Mihaela van der Schaar, *Senior Member, IEEE*

***Abstract*— An increasing number of networked and battery-powered devices require the simultaneous processing of multiple high-complexity dynamic multimedia tasks. Existing resource allocation solutions include both centralized approaches and distributed approaches. Centralized approaches typically have two fundamental limitations. First, multimedia algorithms typically have a large number of operating points and thus requires high implementation complexity to optimally allocate resources across multiple applications, making centralized solutions unsuitable for dynamic environments. Second, autonomous multimedia applications may not be willing to reveal information about their private utility functions, since such information can be exploited by other selfish applications to gain an unfair performance advantage by strategically misusing the system resources. On the other hand, existing decentralized solutions typically solve only resource-constrained utility maximization problems, without regard to other system platform-dependent costs and objectives, and without considering application dynamics. To address these limitations, we present in this paper a decentralized low-complexity resource management solution that can optimize a variety of system performance objectives without requiring applications to provide their utilities. We demonstrate analytically and experimentally that our algorithms converge quickly to their respective optimal solutions and are thus practical in dynamic environments.**

***Index Terms*— Autonomous multimedia applications, decentralized algorithms, information privacy, multitask resource management, resource-aware, system optimization.**

I. Introduction

With the advent of web TV, YouTube, peer-to-peer multimedia streaming, video conferencing, etc., multiple autonomous multimedia processing, compression, transcoding, and streaming tasks need to be executed simultaneously on the same system. Because multimedia systems must readily cope with time-varying resource availabilities and demands, an automated resource management solution for gathering on-the-fly application requirements, and reconfiguring the system in a timely manner, is highly desirable [2], [8]. Moreover, multimedia applications are highly *resource-aware*, such that they can achieve different video/audio quality levels given different amounts of system resources (e.g., processing power, memory, battery power, bandwidth, etc.) [18], [23]. Hence, the principles and methodology for determining how to *optimally* divide limited system resources among multiple multimedia applications must also be addressed [4].

Various analytical solutions have been proposed for allocating limited system resources to multiple resource-aware applications [4]–[8]. In these works, the quality of a resource allocation scheme is assessed by one of several *social welfare* functions, often characterized by a weighted sum or a weighted product of individual application utilities (or multimedia qualities) [3], [9], [15]. In [4]–[6], centralized resource allocation solutions are provided for maximizing the social welfare of multiple applications that have multiple quality-of-service dimensions (e.g., error rate, delay, etc.), and consume multiple types of system resources. In [7], a centralized solution is provided for maximizing the social welfare of multiple applications running on a multiprocessor system. Note that centralized solutions implicitly require a *resource manager* (RM) to gather application utility-resource functions and solve a centralized optimization problem.

On the other hand, a decentralized resource allocation solution is desirable for three reasons. First, multimedia applications are diverse, and include algorithms such as coding, error concealment, resizing, deinterlacing, deblocking, and denoising, which can function at a fine-granular set of operating points based on various modes and parameters [18], [20], [22]. Hence, the RM must be either complex enough to map out and optimize over a large number of system configurations, or intelligent enough to choose only a subset of potentially optimal candidate policies based explicitly on the application's algorithm. Unfortunately, the computational overhead associated with enumerating possibly hundreds or thousands of policies, and choosing the optimal policy (through centralized optimization or decentralized bidding), is unsuitable for multimedia applications with highly *dynamic* utility functions, since the system would need to be reconfigured frequently during runtime [23]. On the other hand, designing an intelligent RM that can provide candidate policies based on multimedia algorithms is an even more daunting task, since the RM needs to be updated every time a new algorithm is ported to the system. While the complexity of determining candidate policies may not be an issue for a system that supports only one or two types of multimedia algorithms, systems nowadays need to support many different types of algorithms (e.g., see [12] for a list of common video codecs), and hence the size of the RM would need to be very large.

Manuscript received June 18, 2008; revised December 6, 2008. Current version published September 16, 2009. This work was supported by the Grants NSF 0509522 and NSF 0541453. This paper was recommended by Associate Editor J. F. Arnold.

B. Foo was with the Department of Electrical Engineering, University of California, Los Angeles, CA. He is now with the Advanced Technology Center, Lockheed Martin Space Systems Company, Sunnyvale, CA, 90095-1594 USA (e-mail: bkungfoo@ce.ucla.edu).

M. van der Schaar is with the Department of Electrical Engineering, University of California, Los Angeles, CA, 90095-1594 USA (e-mail: 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/TCSVT.2009.2022794

Secondly, and perhaps more importantly, multimedia applications that share a system are often developed by many different companies [12]. In many cases, these competing companies may have incentives to develop products that can exploit information from other applications in order to "selfishly" improve their own performances at the cost of social welfare. Hence, multimedia applications often require autonomy over the distribution of *private information*, such as details about their coding algorithms and utilities, in order to protect themselves from selfish applications (and a potentially exploitative system). Unfortunately, the algorithms proposed in [4]–[7] require multimedia applications to submit their utility-resource functions to the RM. Likewise, CARISMA's auction-based protocol requires each application to bid *according to its utility* for each candidate policy, such that each application would be required to report its utility to the middleware for possibly many different system configurations [8]. In summary, these works do not offer *informationally* decentralized solutions that can protect the private information of autonomous multimedia applications.

Finally, multimedia applications are often networked, and hence information such as rate–distortion (R–D) tradeoffs [10], [11], as well as complexity–distortion tradeoffs [18], [23], are known only at the source. Various informationally decentralized optimization solutions have been proposed for networked applications [8]–[11] which maximize the sum of applications' utilities in a decentralized manner. These decomposition approaches enable remote video encoders to make coding decisions in order to satisfy a decoding system's bandwidth constraint or computational resource bounds.[1] However, most of these works address only a single problem: maximizing the sum of utilities for multiple applications under shared resource constraints.

In this paper, we investigate how a system RM with little information about multimedia applications can *arbitrate* resource divisions to achieve a variety of application and system objectives, under application dynamics. Our main contributions are summarized below.

1) We introduce a general analytical framework for the exchange of demands and costs between applications and the RM, which we denote the *message exchange protocol* (MEP). Rather than focusing on a single decentralized solution that maximizes the sum of application utilities (as in prior works), we demonstrate how to use the MEP to implement decentralized algorithms that achieve a wide gamut of system objectives, such as minimizing system energy consumption, performing workload balancing over multiple processing elements, and performing joint power scheduling for interdependent multimedia jobs.
2) In contrast to prior approaches where decentralized optimization is carried over static utility functions, we also consider application-specific dynamic utility functions. By modeling temporal correlations in the quality-resource functions for various video sequences, we determine how the system can improve the rate of adaptation for MEP-based decentralized algorithms, and how the information exchange overhead of the MEP-based algorithms can be reduced. We show experimentally that our proposed algorithms converge quickly to their respective optimal solutions and are therefore ideal for resource adaptation in dynamic environments.

The organization of our paper is as follows. Section II provides an overview of the MEP, and demonstrates how conventional dual decomposition fits within the framework. In Section II, we provide MEP-based algorithms for jointly maximizing social welfare and minimizing energy consumption. Section III provides MEP-based algorithms for achieving several other miscellaneous objectives. Section IV analyzes how parameters in the MEP can be tuned to adapt quickly to time-varying, but temporally-correlated video quality functions. Section V provides simulations that compare the performances of the algorithms, and Section VI concludes our paper.

II. Message Exchange Protocol

In this section, we introduce the analytical framework for the MEP that is used to communicate information between applications and the system, as illustrated in Fig. 1. As an illustrative example, we demonstrate how the social welfare maximization problem [9] can be solved using the proposed MEP.

A. Message Exchange Protocol

The focus of our paper is to design a resource management solution that can provide optimal resource allocation schemes for multiple multimedia applications in an informationally decentralized environment. We consider an RM that can accurately monitor resource utilization across possibly multiple processors or machines.[2] Moreover, the RM can collect statistics about the computational requirements from various tasks. However, the RM is not given each applications' internal utility functions, since such information is part of the companies' intellectual property, and can possibly be exploited by other competing applications. Hence, the RM cannot schedule resources in a centralized manner to maximize objectives that include social welfare-related metrics (e.g., sum of application utilities).

While the RM does not have knowledge of application utilities, it makes use of a continuous model of system resources to intelligently effect various resource allocation solutions by charging applications various *costs* for consuming system resources. The RM generates costs in the form of *tax functions*, messages that capture the congestion levels or consumption rates of various utilized system resources (e.g., CPU utilization, communications bandwidth, energy availability, etc.). Based on the tax functions provided by the RM,

[1]Note that while the utility maximization framework involves a *continuous* representation of system resources, a continuous quality-complexity function is a good approximation for fine-grained scalable multimedia applications. Furthermore, while each application is required to perform individual utility optimizations, many multimedia applications already have features built in to maximize their qualities subject to resource constraints [23].

[2]For example, in a multiple processor system, the RM can be located in the OS kernel, which handles scheduling and resource allocation.

multimedia applications can factor the costs into their utilities, reconfigure their algorithms, and update their resource requirements, which effectively leads to a new system configuration. The message exchange protocol (MEP) is presented formally as follows:

Stage 1: Initialization.

The RM provides each task i, $i \in \{1, 2, \ldots, I\}$, with a parametrized tax function $t_i(\mathbf{x}_i, \mathbf{m}_{-i})$, where $\mathbf{x}_i$ is a vector of task i's resource demands, and $\mathbf{m}_{-i}$ is a parameter (or *message*) that is transmitted and updated by the RM.[3] The message can be used to communicate the system resource condition, such as CPU utilization, system energy consumption, bandwidth availability, etc. Upon initialization, the RM also transmits an initial message $\mathbf{m}_{-i}^{(0)}$.

Stage 2: Message Exchange.

During the message exchange phase, the RM and the tasks will iteratively perform optimizations and message exchanges (as shown below) until the resource allocation scheme converges. x *Tasks to RM:* During iteration n, each task i solves the following local optimization problem:

$$\mathbf{x}_i^{(n)} = \arg\max_{\mathbf{x}_i} \left\{ Q_i(\mathbf{x}_i) - t_i\left(\mathbf{x}_i, \mathbf{m}_{-i}^{(n-1)}\right)\right\} \tag{1}$$

where the *net utility* function of task i is the task quality function $Q_i(x)$ minus the tax $t_i\left(\mathbf{x}_i, \mathbf{m}_{-i}^{(n-1)}\right)$ that the system charges to the task. After computing its new resource demand vector $\mathbf{x}_i^{(n)}$, it submits $\mathbf{x}_i^{(n)}$ to the RM.
RM to Tasks: Based on the resource demands from all tasks, the RM relays messages $\mathbf{m}_{-i}^{(n)} = f_i\left(\mathbf{x}_{-i}^{(n)}, \gamma^{(n)}\right)$ back to each task i, where $\mathbf{x}_{-i}^{(n)}$ is a vector of the resource demands from all other tasks during iteration n, $\gamma^{(n)}$ is a step size parameter that affects the value of the message at every iteration, and f_i is a function that is designed in a way to provide sufficient information to each task while minimizing the amount of information exchanged (i.e., the RM may not need to transmit the entire demand vector $\mathbf{x}_{-i}^{(n)}$ and parameter $\gamma^{(n)}$ to each task i). Minimizing the message size can be beneficial both in terms of reducing the informational overhead, as well as better protecting information about individual applications' resource demands. An example of messages $m_{-i}^{(n)}$ will be given in the next subsection.

Stage 3: Task Resource Allocation.

For static environments, *convergence* of the MEP to an optimal resource allocation solution $\mathbf{x}^* = \left[\mathbf{x}_1^{*T}, \ldots, \mathbf{x}_I^{*T}\right]^T$ is highly desirable. Formally, convergence can be defined as follows: For any $\varepsilon > 0$, there exists an integer N such that the resource allocation vector at any iteration $n \geq N$, given by $\mathbf{x}^{(n)}$, satisfies the property $\left\|\mathbf{x}^* - \mathbf{x}^{(n)}\right\|_2 < \varepsilon$ [19]. In practice, an approximate convergence point can be chosen for any given ε by terminating the algorithm when the change in the resource demand vector is less than ε, i.e., $\left\|\mathbf{x}^{(n)} - \mathbf{x}^{(n-1)}\right\|_2 < \varepsilon$. After terminating the MEP, each task i is allocated $\mathbf{x}_i^{(n)}$ resources.

[3]The tax charged to each task may be in the form of real money or tokens for future system usage. As such, it makes sense to define $t_i\left(0, \mathbf{m}_{-i}\right) = 0$, such that a task does not pay if it chooses not to utilize system resources.

For dynamic environments on the other hand, a highly desired objective is minimizing the *adaptation time*. The adaptation time can be defined as the number of iterations for the algorithm to provide a solution within ε of the optimal resource allocation, given that the application utility functions are time-varying. Importantly, unlike the convergence time in a static environment, the algorithm must constantly adapt to different application utility functions over time. However, the RM can also take advantage of temporal correlations in the utility functions to reduce the adaptation time (discussed in more detail in Section V).

We note that, in general, MEP-based algorithms will require *multiple* iterations to converge due to a lack of centralized utility information. Furthermore, after each iteration, the resource allocations may change, such that resulting resource allocations may become temporarily infeasible [25]. However, feasibility can be achieved by projecting the solution onto the feasible set (e.g., scaling down the resource allocations proportionally), although the resulting allocation will often be suboptimal. A more viable solution would be to perform iterations of the MEP prior to the transmission of a video scene or group of frames (GOP), where the MEP can be run in the background during transmission of the prior scene/GOP and allowed to converge. (This will be discussed in more detail in the simulations section.) Finally, note that the MEP requires the quality-resource function of each application to be concave. Concavity can always be guaranteed for multimedia applications, since each application can generate a piecewise linear convex hull using Pareto optimal quality-resource operating points, as shown in [20], [21]. Based on the concavity of quality-resource functions, each iteration of the MEP requires each multimedia application to solve only a low-complexity convex optimization problem [19].

B. Illustrative Example of MEP Implementation: Dual Decomposition

We now demonstrate that dual decomposition falls within our MEP framework. Consider an objective function that is the sum of the video dB PSNR qualities $Q_i(x_i)$ [8], [15], [16]

$$U(\mathbf{x}) = \sum_{i=1}^{I} Q_i(\mathbf{x}_i) \tag{2}$$

where vector $\mathbf{x}_i$ indicates the amount of resources (bandwidth, computational resources, etc.) used by task i. The social welfare maximization (SWM) problem can be given by

$$\begin{aligned} &\max \sum_{i=1}^{I} Q_i(\mathbf{x}_i) \\ &\text{s.t. } g_j(\mathbf{x}) \leq R_j,\ j = 1, \ldots, J, \\ &\mathbf{x}_i \geq 0\ \forall i \in I \end{aligned} \tag{3}$$

where each $g_j(\mathbf{x}_1, \ldots, \mathbf{x}_I) \leq R_j$ is a (convex) system constraint. For example, in multiuser video transmission, $\mathbf{x}_i$ is a scalar that expresses the rate of user i, and a single bandwidth constraint $g(\mathbf{x}) = \sum_{i=1}^{I} \mathbf{x}_i \leq R$ is used [11]. A similar formulation can be used for computational resource constraints on a shared system [7]. The tax function in the MEP protocol

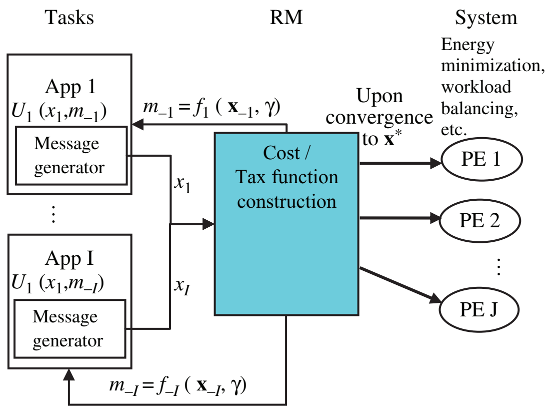

Tasks
RM
System
Energy minimization, workload balancing, etc.
App 1
$U_1(x_1, m_{-1})$
Message generator
$m_{-1} = f_1(\mathbf{x}_{-1}, \gamma)$
x_1
Cost / Tax function construction
Upon convergence to $\mathbf{x}^*$
PE 1
PE 2
PE J
App I
$U_1(x_1, m_{-I})$
Message generator
x_I
$m_{-I} = f_{-I}(\mathbf{x}_{-I}, \gamma)$

Fig. 1. Message exchange protocol for resource allocation.

can be given by $t_i\left(\mathbf{x}_i, \mathbf{m}_{-i}^{(n-1)}\right) = \left(\mathbf{p}^{(n-1)}\right)^T \mathbf{x}_i$, where $\mathbf{p}^{(n-1)}$ is a cost vector per unit resource (e.g., cost per CPU cycle, unit bandwidth, etc.) charged to each task. During each iteration n, each task $i \in \{1, 2, \ldots, I\}$ calculates its new resource demand $\mathbf{x}_i^{(n)}$ based on

$$\mathbf{x}_i^{(n)} = \arg\max_{\mathbf{x}_i \geq 0} \left\{ Q_i(\mathbf{x}_i) - \left(\mathbf{p}^{(n-1)}\right)^T \mathbf{x}_i \right\} \tag{4}$$

and submits it to the RM. The RM then updates the cost vector $p^{(n)}$ by

$$\mathbf{p}^{(n)} = \mathbf{p}^{(n-1)} + \left(\mathbf{g}\left(\mathbf{x}^{(n)}\right) - \mathbf{R}\right) \Big/ \gamma^{(n)} + \chi_+\left(\mathbf{x}^{(n)}, \gamma^{(n)}\right) \tag{5}$$

where $\mathbf{g}(\mathbf{x}) = [g_1(\mathbf{x}), \ldots, g_J(\mathbf{x})]^T$, $\gamma^{(n)}$ is the step size parameter that affects the change in the cost at each iteration, and $\chi_+\left(\mathbf{x}^{(n)}, \gamma^{(n)}\right) = \max\left(\mathbf{0}, \left(\mathbf{g}\left(\mathbf{x}^{(n)}\right) - \mathbf{R}\right)/\gamma^{(n)}\right)$ is a metric to force the cost to be nonzero if the cost is initialized at zero, and resource demands are infeasible [25]. The RM submits messages $\mathbf{m}_{-i}^{(n)} = \mathbf{p}^{(n)}$ back to each task $i \in \{1, \ldots, I\}$.

After the messages have been exchanged repeatedly, and the resource demands converge to a feasible vector $\mathbf{x}^*$, each task i is then allocated resource quantity $\mathbf{x}_i^*$. It has been shown that convergence can be guaranteed if the value of $\gamma^{(n)}$ is picked at every stage of the process from an increasing sequence, as long as the sum of the sequence generated by $1/\gamma^{(n)}$ is infinite (e.g., $\gamma^{(n)} = n$) [24], [25].

III. MEP-Based Algorithms for Energy Minimization

A. Energy Minimization for an Always Active System

Now that we have shown that the dual decomposition method falls within our MEP framework, in the following sections we demonstrate how the MEP framework can be utilized by the system RM to compel applications to demand resources in a manner that converges to other system objectives. In this section, we propose an MEP-based algorithm to simultaneously maximize the social welfare of applications while minimizing energy consumption for dynamic voltage scalable (DVS) systems. A DVS-enabled processor can change its operating frequency by adjusting its voltage level, which affects its energy consumption rate. The active energy consumed by a DVS-enabled processor can be modeled as a convex increasing function of the processor workload [13], [14], [17]. The total energy consumption can be modeled as a convex function of the *total workload* across the system.[4] The multiobjective optimization problem can be given by

Social Welfare-Max. Energy-Min (SWMEM) Problem

$$\begin{aligned} &\max_{x_i} \textstyle\sum_{i=1}^{I} Q_i(x_i) - \lambda E_{\text{tot}}^{\text{act}}\left(\sum_{i=1}^{I} x_i\right) \\ &\text{s.t. } x_i \geq 0 \end{aligned} \tag{6}$$

where x_i denotes the total computational resources allocated to task i, $E_{\text{tot}}^{\text{act}}(R)$ is the minimum *active* energy of a multiprocessor system given the total computational resources allocated R, and $\lambda > 0$ is a weighting factor that determines the relative importance of system energy to social welfare. Note that the *passive* energy is assumed to be constant and therefore not considered in the equation. However, the passive energy becomes important for multiprocessor systems where each of the multiple processing elements (PEs) can enter sleep mode, as discussed in the next section. The MEP tax function that achieves the optimal solution to (6) is

Excess Energy Minimizing (EEM) Tax Function

$$t_i(x_i) = \lambda E_{\text{tot}}^{\text{act}}(x_i + d_i) - \lambda E_{\text{tot}}^{\text{act}}(d_i) \tag{7}$$

where x_i is task i's total computational resource demand, and d_i is task i's "perceived" total computational resource demand from all other tasks (which will be discussed later). The EEM tax function has the following interpretation: The system charges each task the amount of energy the system consumes when the task runs on the system, minus the amount of energy the system consumes if the task does not run on the system. Note that the second term in the tax function, $\lambda E_{\text{tot}}^{\text{act}}(d_i)$, does not depend on x_i, but is there to ensure that each task i is not taxed if it does not consume system resources, i.e., $t_i(0, m_{-i}) = 0$.

As in the SWM algorithm, the RM can introduce a step size $\gamma^{(n)}$ to ensure convergence of the EEM algorithm. In this case, assuming that $x_i^{(n-1)}$ is the computational demand from each task i during the previous iteration, and $x_i^{(n)}$ the demand during the current iteration, the RM submits the following message to task i to update the tax function:

$$d_i = m_{-i}^{(n)} = \sum_{l \neq i} \left(x_l^{(n-1)}\right) + \frac{1}{\gamma^{(n)}} \sum_{l \neq i} \left(x_l^{(n)} - x_l^{(n-1)}\right) \tag{8}$$

such that the "change in demand" seen by each task is scaled down by $\gamma^{(n)}$. The step size parameter $\gamma^{(n)}$ can be chosen from an increasing sequence to guarantee convergence. We omit a rigorous proof of convergence here, as a similar proof can be found in [15], but we will prove that for strictly concave task quality functions (in terms of x_i) and convex energy functions, the EEM tax function generates an optimal solution for the SWMEM objective upon convergence. This is an important result for multimedia applications, where the quality-complexity functions are concave [18]. The convergence time will be further analyzed in (Section VI).

[4]We note that this is an incomplete model, since the energy function may not be purely convex, or purely a function of the total workload across multiple cores. Future work can explicitly address overheads associated with processor sharing and memory access.

Proposition 1: For strictly concave quality functions and convex energy functions, the EEM tax function generates an optimal solution to the SWMEM upon convergence.

Proof: See Appendix. We note that the proof provides us with an additional guideline for designing tax functions that achieve globally optimal solutions, i.e., tax functions must be designed such that at equilibrium, the Karush–Kuhn–Tucker (KKT) conditions are uniquely and simultaneously met for each task's local optimization problem as well as the system's global optimization problem.

B. Energy-constrained SWM for Multiprocessor Systems With Sleep Modes

In this section, we consider a social welfare-maximizing solution for a multiprocessor system with PEs that are DVS-enabled and can switch between active and sleep modes. We assume that a PE can go to sleep during a control interval of length T and consume negligible energy if it is not assigned any jobs to process. However, a PE cannot be both active and sleeping within the same control interval. For an active DVS-enabled PE j, it is shown that, given a total computation requirement (e.g., cycles) r_j during an interval of length T, the PE should run at a constant frequency $f_j = r_j/T$ throughout the interval in order to minimize energy consumption [17]. Thus, the sleep mode-enabled energy function for each PE j can be given by

$$E_j\left(r_j\right) = \begin{cases} 0, & r_j = 0 \\ TP_j^{\text{pass}} + \alpha_j TP_j^{\text{act}}\left(\frac{r_j}{T}\right), & r_j > 0 \end{cases} \tag{9}$$

where the passive power for PE j, P_j^{pass}, is constant, and the active power $P_j^{\text{act}}\left(f_j\right)$ is a convex function of the operating frequency f_j [13], [14].[5]

Note that the energy function for each PE is no longer convex due to the discontinuity at $r_j = 0$ in (9). However, because the function is piecewise convex, we can still identify all the local minima in the multiprocessor energy function, and search through them in a combinatorial fashion to find the globally optimal point. Prior to running the SWM, we determine, for a given energy constraint, the subset of PEs to turn on in order to maximize the available computational resources. The optimal configuration is the one with the loosest computational resource constraint. The algorithm is summarized in Table I.

TABLE I
SLEEP MODE INITIALIZATION FOR SWM

```
1.  // Initialize before runtime.
2.  **Fix** total energy constraint $E_{\text{tot}}$.
3.  **Set** $S = \{1, 2, \ldots, J\}$ to be the set of PEs.
4.  **Construct** the *power set* (set of all possible subsets) of PEs, $2^S$.
5.  **For** each subset of PEs $N \in 2^S$
6.      **Calculate** the active energy constraint: $E_{\text{tot}}^{\text{act}} = E_{\text{tot}} - \sum_{j\in N} E_{\text{pass}}^{N}$.
7.      **Determine** the computational resource constraint $\Upsilon_N = R_N\left(E_{\text{tot}}^{\text{act}}, N\right)$ by using the inverse of the energy model $E_{\text{tot},N}^{\text{act}}(R)$ for only the PEs in $N$.
8.  **End For**
9.  **Turn on** the PEs corresponding to the $N$ with the largest $\Upsilon_N$.
10. // During runtime.
11. **Run SWM** with complexity constraint $\Upsilon_N$.
```

IV. MEP-Based Algorithms for Miscellaneous System Objectives

A. Assigning Tasks to PEs Using Tax Functions

A practical concern for multiprocessor systems is how to assign tasks to different PEs in order to minimize energy consumption. Many multicore systems provide software programmers with the flexibility to choose system configurations that can better exploit thread-level parallelisms in their applications [1], [2]. While this can improve the performance of individual parallelizable applications when resource availability is high, arbitrary resource usage can hurt the overall performance of an energy-constrained system, since many applications might compete over the same set of over-utilized PEs. One possible solution is to remove this programming flexibility and allow the RM to centrally schedule applications across the PEs. However, we propose an alternative idea which keeps the flexible programming feature, but introduces tax functions to *compel* applications to run on PEs that are either underutilized, or have lower energy consumption rates. We note that determining this optimal assignment of PEs is beyond the scope of the paper. The main purpose of this section is to provide an alternative practical method for the RM to assign applications to PEs, given that an efficient assignment has already been computed.

To provide an illustration of our algorithm, we choose a tax function that compels individual tasks to run on individually assigned PEs. If a task is assigned to a particular PE, it is taxed on the amount of energy consumed as if no other tasks were running on the PE. However, if a task is not assigned to a particular PE, it must pay a penalty equal to the total energy increase it causes to all other tasks sharing that PE. Without loss of generality, we assign task i to a PE $j = i$, where we have assumed that $I \leq J$, such that every task has a uniquely assigned PE. If task i makes resource demands across J PEs given by $\mathbf{r}_i = \left(r_{i,1}, r_{i,2}, \ldots, r_{i,J}\right)$, its quality is modeled by a concave increasing function $Q_i\left(\sum_{j=1}^{I} r_{i,j}\right)$. Using an appropriate step size $\gamma^{(n)}$, the RM sends the following messages to each task i during each iteration n:

$$m_{-i}^{(n)} = \left(d_{i,1}, d_{i,2}, \ldots, d_{i,i-1}, d_{i,i+1}, \ldots, d_{i,J}\right)$$
$$\text{where } d_{i.j} = \sum_{l\neq i}\left(r_{l.j}^{(n-1)}\right) + \frac{1}{\gamma^{(n)}}\sum_{l\neq i}\left(r_{l,j}^{(n)} - r_{l,j}^{(n-1)}\right) \tag{10}$$

where $r_{i,j}^{(n)}$ is task i's resource demand on processor j during iteration n. The application then maximizes its utility based

[5]Note that when there are finitely many voltage levels for a PE, the power as a function of CPU requirements is piecewise continuous and increasing. Since energy consumption is the integral of power with respect to time, it follows that energy is a piecewise linear (and convex) function due to the power being a nondecreasing function.

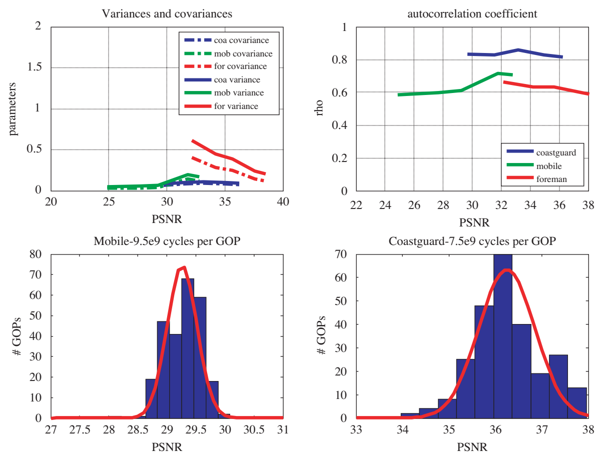


Fig. 2. (a) Variances and covariances and (b) autocorrelation coefficient ρ for quality perturbations in the *Coastguard*, *Mobile*, and *Foreman* sequences. (c–d) Gaussian fits to the PSNR distributions (in dB) for *Mobile* and *Coastguard* when operating at particular complexity levels. The complexity is the number of entropy decoding and inverse transform cycles per GOP.

on the following tax function:

The PE Assigning Tax Function (PA):

$$t_i^{\mathrm{PA}}\left(\mathbf{r}_i, m_{-i}^{(n)}\right) = \alpha_i E_i^{\mathrm{act}}\left(r_{i,i}\right) + \sum_{j\neq i} E_j^{\mathrm{act}}\left(r_{i,j}+d_{i,j}\right) - \sum_{j\neq i} E_j^{\mathrm{act}}\left(d_{i,j}\right). \quad (11)$$

Here, task i's perceived total resource demand on PE j by all other tasks is given by $d_{i,j}$. Similar to the EEM tax function, the perceived change in demand is scaled by step size $1/\gamma^{(n)}$.

The PA algorithm satisfies the condition where a task does not have to pay taxes for any processor that it does not use. It can also be shown that the decentralized PA algorithm often converges to a solution where a task will rarely run on another task's assigned processor, unless the derived benefit is very large. (The proof is similar to the Appendix and is therefore omitted.) Note that while we have constructed a tax function to assign a single processor to each task in this section, a similar tax function exists for assigning multiple processors to each task. This can be identically achieved by using the same type of reduced taxation for a *set* of dedicated processors.

B. Power Scheduling for Interdependent Multimedia Jobs

In this section, we propose a power scheduling solution for maximizing the social welfare of multiple applications with interdependent jobs. For example, in MPEG, if an I-frame is decoded at a lower resolution using less resources, the following P-frames and B-frames will also be decoded under extra distortion regardless of the amount of resources allocated to them, since they depend on the distorted I-frame to reproduce their respective video frames. Hence, proper power scheduling is an important issue for optimizing the performance of real-time multimedia systems where jobs do not only have stringent delay deadlines, but the contribution of jobs to the overall quality may also be highly interdependent.

We formulate the scheduling problem as follows. Define a super-interval T_S as consisting of N_S time intervals of size T, where T is the time between successive video frames, and N_S is the size of an interdependent group of frames/pictures (GOP). For the mth time interval in super-interval T_S, a corresponding quantity of computational resources $x_{i,m}$ is allocated to task i. The resulting quality function with N_S time intervals per super-interval is

$$Q_i\left(\mathbf{x}_i\right) = Q_i\left(x_{i,1}, x_{i,2}, \ldots, x_{i,N_S}\right) \quad (12)$$

where $\mathbf{x}_i = \left[x_{i,1}, x_{i,2}, \ldots, x_{i,N_S}\right]^{\mathrm{T}}$. For simplicity, we allow only the time interval $[(m-1)T, mT)$ in the super-interval for decoding the mth job in the GOP.

The system objective is then to simultaneously maximize the sum of task qualities while saving energy by jointly allocating resource shares to each task, and adjusting the operating level of the DVS-enabled processor for each time slot. The optimization function is as follows:

Joint SWMEM and Scheduling Optimization Problem (SWMEM-S):

$$\max_{\mathbf{x}_i} \sum_{i=1}^{I} U_i\left(\mathbf{x}_i, \lambda\right) = \max_{\mathbf{x}_i} \sum_{i=1}^{I} Q_i\left(\mathbf{x}_i\right) - \lambda \sum_{m=1}^{N_S} E_{\mathrm{tot}}^{\mathrm{act}}\left(\sum_{i=1}^{I} x_{i,m}\right) \quad (13)$$
$$\text{s.t. } \mathbf{x}_i \geq 0.$$

Since SWMEM-S is simply the multidimensional version of the SWMEM, by a proof similar to Proposition 1, we can show that the MEP tax function which achieves this global objective function is the corresponding multidimensional version of the EEM.

Joint EEM and Scheduling Tax Function (EEM-S):

$$t_i^{\mathrm{EEM-S}}\left(x_{i,m}, d_{i,m}\right) = \lambda \sum_{m=1}^{N_S}\left[E^*\left(x_{i,m}+d_{i,m}\right) - E^*\left(d_{i,m}\right)\right] \quad (14)$$

where $d_{i,m} = \sum_{l\neq i}\left(x_{i,m}^{(n-1)}\right) + \left(1/\gamma^{(n)}\right)\sum_{l\neq i}\left(x_{i,m}^{(n)} - x_{i,m}^{(n-1)}\right)$ is computed by the RM from the resource demands $x_{i,m}^{(n)}$ for each interval m during algorithm iteration n.

C. Summary of MEP Algorithm Design Requirements

The examples in Sections III and IV were provided to illustrate a formal method for decentralized algorithm design via tax functions. To clarify our approach, three main requirements are summarized below.

First, an analytical optimization problem must be defined. Are we maximizing the social welfare or minimizing system resource usage (energy, workload on specific PEs, etc.)? What are the resource constraints? Secondly, a proper step size sequence $\gamma^{(n)}$ must be chosen to guarantee convergence in static environments [24]. However, as will be discussed in the next section, different choices of $\gamma^{(n)}$ can also affect the adaptation time of decentralized algorithms in dynamic environments. Finally, the KKT conditions must be simultaneously and uniquely met for the system objective function and each task's local optimization function (e.g., tax function) to guarantee optimality.

V. Efficient Resource Allocation for Temporally Correlated Quality Functions

In this section, we propose adaptive resource allocation solutions that explicitly take advantage of the temporal correlations exhibited by the time-varying quality-resource functions of multimedia applications. For illustration purposes, we focus on the MEP-based SWM algorithm for video decoding applications. First, we propose a Markov model for video decoding quality functions based on the collected statistics from various video sequences (Section V-A). We then discuss the information required by the RM to minimize the adaptation time, and we present how to choose statistically optimal step sizes for the SWM algorithm based on information known about the dynamics of the video quality functions (Section V-B). Finally a simple algorithm with very low informational overhead is proposed (Section V-C), which fixes the cost function during intervals where video quality functions do not vary significantly.

A. Modeling the Quality Functions for Dynamic Video Applications

Video sequences exhibit highly time-correlated characteristics, and are often encoded using similar GOP structures over time. Hence, we can model video decoding quality-resource functions as discrete time Markov random processes $Y_i(x_i, n)$, where each index n corresponds to the GOP number in the sequence, and the computational resource demand x_i in each period n corresponds to the complexity of decoding the entire GOP. (Note that we use n to index each GOP here, while it was previously used as the iteration number in our MEP framework. We will show through simulations that, in fact, these two are equivalent indices, since, only a single iteration is required between adjacent GOPs to achieve near-optimal performance!) The characteristics of the Markov process, such as its distribution, mean, or variance, can be obtained by each application using a sliding window, or can be estimated by looking up a table of classified sequences for the specific coder. For example, the distribution of video decoding quality for several video sequences are given in Fig. 2 using the coder in [18]. We found that the Gaussian distribution approximates well the PSNR statistics (in dB) of each sequence for fixed complexity levels [Fig. 2(c) and (d)]. Hence, we model the video decoding quality function for each task i as a Gaussian Markov chain. The Gaussian Markov chain can be perfectly described by the mean quality function $Q_i(x_i)$, the perturbation variance $\sigma_i^2(x_i)$, and the autocorrelation coefficient ρ_i between adjacent GOPs in the sequence [See Fig. 2(a) and (b).], i.e.,

$$\begin{aligned} Y_i(x_i, n) &= Q_i(x_i) + \sqrt{\rho_i}\,(Y_i(x_i, n-1) - Q_i(x_i)) \\ &\quad + \sqrt{1-\rho_i}\,W_i(x_i) \\ &= \left(1-\sqrt{\rho_i}\right) Q_i(x_i) + \sqrt{\rho_i}\,Y_i(x_i, n-1) \\ &\quad + \sqrt{1-\rho_i}\,W_i(x_i) \end{aligned} \tag{15}$$

where $W_i \sim N\left(0, \sigma_i^2(x_i)\right)$ is a white Gaussian vector with variance $\sigma_i^2(x_i)$. Note that according to our measured statistics [Fig. 2(a) and (b)], ρ_i depends little on x_i, and can therefore be modeled as a constant rather than a function of the complexity. Hence, based on any complexity level x_i, ρ_i can be estimated by

$$\rho_i = \frac{E\left[(Y_i(x_i, n) - Q_i(x_i))\,(Y_i(x_i, n-1) - Q_i(x_i))\right]}{\sigma_i^2(x_i)}. \tag{16}$$

The quality function $Y_i(x_i, n)$ will change its shape based on the first derivative of $\sigma_i(x_i)$. We approximate this perturbation variance using the following linear model

$$\sigma_i(x_i) = a_i x_i + b_i. \tag{17}$$

Hence, the change in the first derivative of the quality function from the previous time interval is

$$\varsigma_i(n) = \frac{dY_i(x_i, n)}{dx_i} - \frac{dY_i(x_i, n-1)}{dx_i} \tag{18}$$

where $\varsigma_i(n)$ is a Gaussian random variable with standard deviation on the order of $\sqrt{\rho}a_i$, which is independent of x_i based on the model in (17). Note that for the SWM algorithm, the larger the variance of $\varsigma_i(n)$, the more suboptimal the allocation will be if the cost per unit resource p is not accurately updated.

B. Informational Requirement for Single Iteration Adaptation, and Optimal SWM Step Sizes

In general, if the RM knows the second derivative $Q_i''(x_i)$ of each task's quality function, the optimal solution can be determined by the following two steps. First, the RM can project a cost per unit resource for the tasks, and each task's response $\hat{x}_i$ can be used to reconstruct the first derivative function given $Q_i'\left(\hat{x}_i\right) = p$. Once $Q_i'(x_i)$ is known, the optimal cost can be obtained, and a centralized solution can be used for resource allocation. Hence, the second derivative provides the RM with sufficient information to optimally update the resource allocation in a single iteration.

However, $Q_i''(x_i)$ is generally not known by the RM. Nevertheless, if some information about $Q_i''(x_i)$ can be gathered by the RM, better convergence time can be guaranteed for the SWM by dynamically adjusting the step size parameter γ (Note that the step size is no longer chosen from a predetermined increasing sequence $\gamma^{(n)}$, but is updated based on the behavior of the video quality functions.) To demonstrate our approach, we propose the following modification to the SWM algorithm:

SWM for Updating Resource Allocations (SWM-U): Based on the model in (15) and (18), we have the following task level optimization function for time interval n:

$$\begin{aligned} &\max_{x_i, p_i} Y_i(x_i, n) - t_i \\ &\quad = \max_{x_i, p_i} Y_i(x_i, n-1) + \varsigma_i(n)\,x_i - t_i(x_i, m_{-i}) \\ &\quad = \max_{x_i, p_i} Y(x_i, n-1) + \varsigma_i(n)\,x_i - p x_i. \end{aligned} \tag{19}$$

Since the costs are already assumed to be nonzero in the previous iteration, we exclude the last term $\chi(\cdot)$ from (5). As can be seen, the random quality perturbation function $W_i(x_i)$ causes the equilibrium point to shift, as the perturbations now

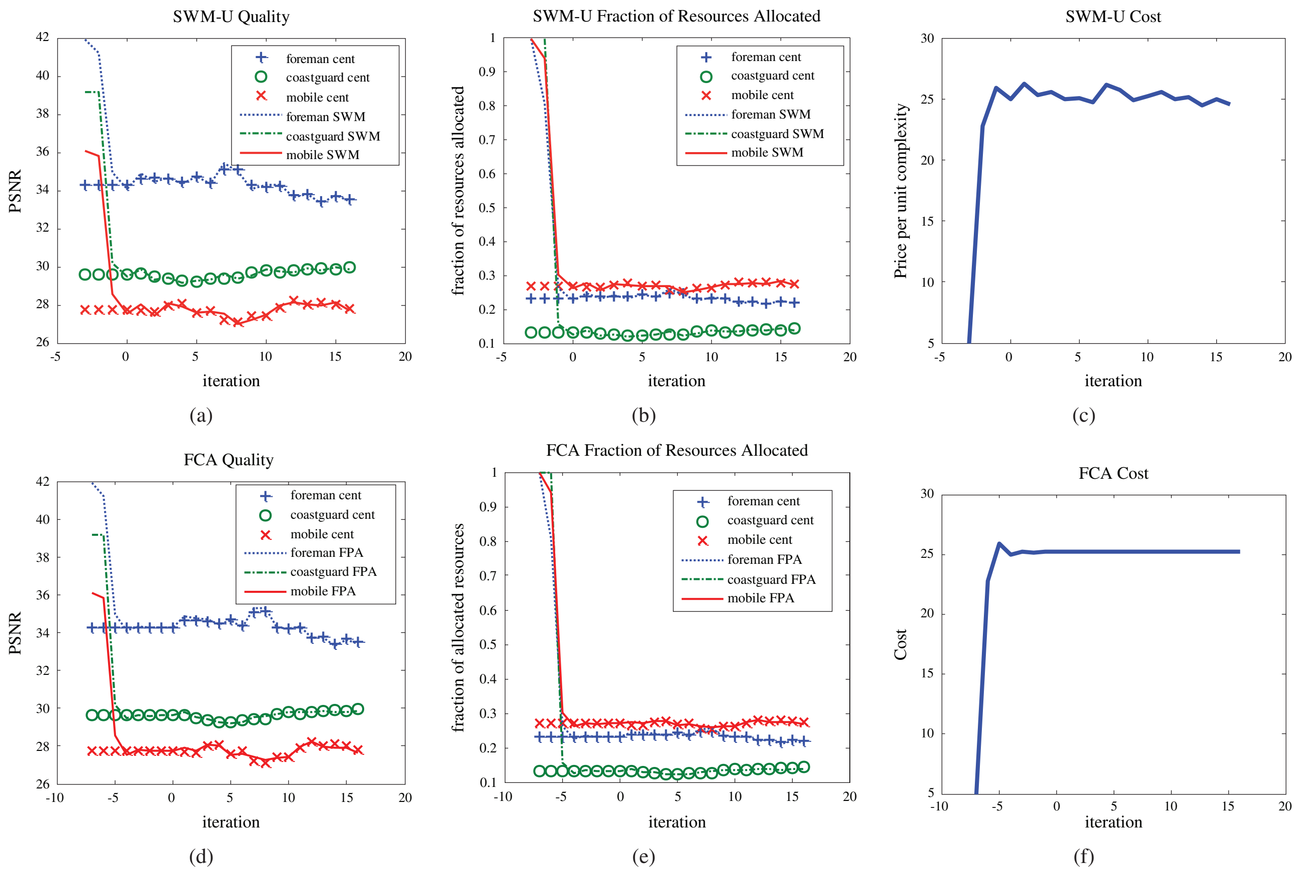


Fig. 3. (a) Convergence of decentralized allocation qualities to centralized solutions, (b) corresponding fraction of resources allocated to each task, and (c) costs are shown for SWM-U. (d–f) Corresponding plots for the fixed-cost algorithm (FCA).

affects each task's perceived cost per unit resource to be $p-\varsigma_i$. To construct a single iteration algorithm that accurately updates resource allocation, a good choice of γ is needed. The following toy example provides insight on how to determine the best choice for γ.

Consider the case when all tasks have identical quality functions $Q(x)$, with initially balanced resource allocations x. At the next time interval, each task identically updates his function to $Q_{\text{new}}(x) = Q(x)+\varsigma x+\xi$, where $\varsigma, \xi \in \mathbb{R}$. Hence, the new derivative can be written as

$$Q'_{\text{new}}(x) = Q'(x) + \varsigma. \tag{20}$$

Based on this new function and the old cost $p = Q'(x)$, a task will update its resource demand to x', where $Q'_{\text{new}}\left(x'\right) = Q'\left(x'\right) + \varsigma = Q'(x)$. Each task now has an identical excess demand of $\Delta x = x' - x$. The second derivative of $Q\left(x\right)$ can be approximated between the points x and x' as

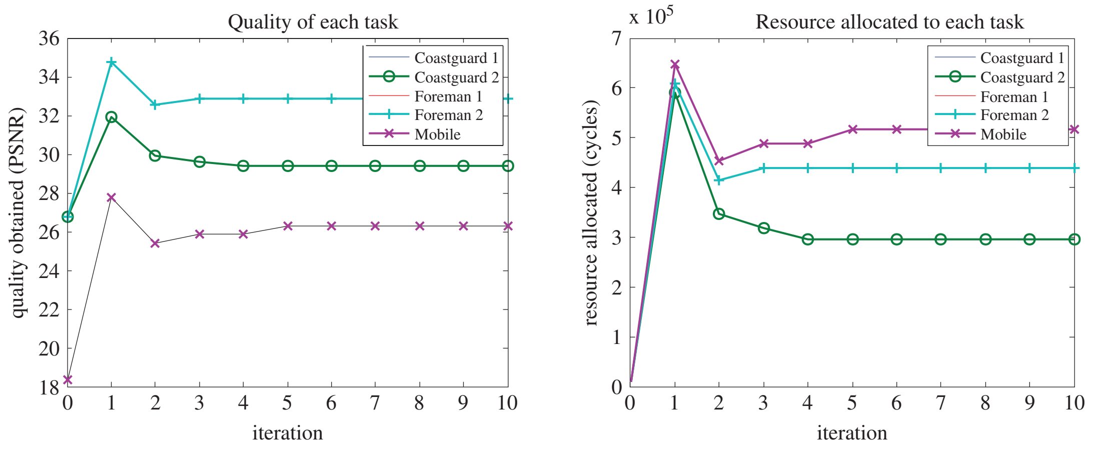
$$Q''(x) \approx \frac{Q'\left(x+\Delta x\right) - Q'(x)}{\Delta x} = \frac{Q'\left(x'\right) - Q'(x)}{x'-x} = \frac{-\varsigma}{\Delta x}. \tag{21}$$

Based on the modified SWM-U tax form (19), setting $\gamma = -\varsigma/\Delta x$ is the optimal choice for that time interval. Note that γ depends on ς and the excess demand; hence if ς is known for every interval, γ can be chosen close to optimal. For the general case with many tasks and different quality-resource functions, γ can be set to the average value of all individually calculated γ_i's.

TABLE II
SWM-U ALGORITHM

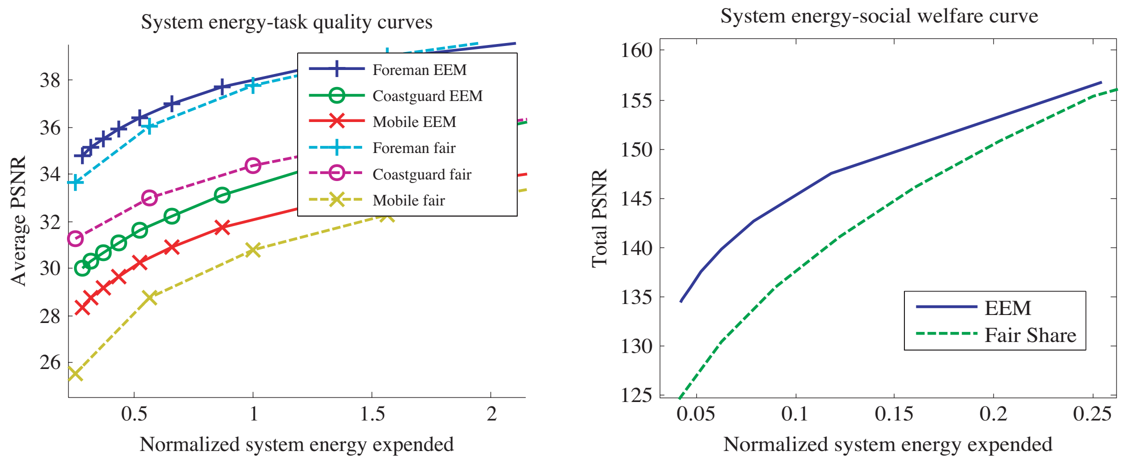
```
1. **Set** initial step size $1/\gamma$.
2. **Run SWM** using an initial price $p$ per unit resource.
3. Obtain resource demands for all tasks $x_1, \ldots, x_I$.
4. **While** tasks are running
5.    **Run SWM** with updated price $p$ using step size $1/\gamma$.
6.    **System RM calculates** $|\Delta x_i|$ for all tasks (using sliding window,
      discounted average, etc.).
7.    **Applications calculate** average $\varsigma_i$ over prior intervals. Transmits
      the parameter to the RM.
8.    **RM finds optimal step size** $\gamma \approx \mathrm{E}\left[\varsigma_i/\Delta x_i\right]_{i=1,\ldots,I}$.
9. **End While**
```

Because the ς parameter depends on changes in the utility-complexity function, it is often known only at the source. Nevertheless, the source can still obtain an estimate of the quality perturbation autocorrelation coefficient ρ and variance σ over several prior time intervals, and use this distribution to determine the mean value of ς. By estimating the excess demand Δx over several intervals, a statistically optimal $\gamma \approx \mathrm{E}\left[\varsigma/\Delta x\right]$ can be obtained. The algorithm for the SWM-U is listed in Table II.



(a) (b)

Fig. 4. (a–b) Convergence of quality and computational resource allocation of EEM with energy function $E \propto x^2$.



(a) (b)

Fig. 5. (a) Energy–quality curves for video sequences in the EEM algorithm compared against the fair share scheduling algorithm. (b) Plot of the sum of qualities (social welfare).

C. Fixed Cost Algorithm

Finally, we propose a low-complexity *fixed cost algorithm* (FCA), which updates resource allocations *without* any communications overhead aside from determining an initial cost per unit resource p. The assumption is that when ς is small, the cost of the SWM-U remains approximately fixed. Hence, the only overhead associated with this algorithm occurs when source characteristics change significantly, and a new cost needs to be determined using a few iterations of the SWM. Otherwise, each task can run the FCA in isolation.

In the FCA, based on the initial cost p obtained by the SWM at the beginning of a video scene, tasks simply determine their demand for each time interval n using the cost p without communicating with the RM, i.e.,

$$x_i^{\text{demand}}(n) = \arg\max_{x_i} Y(x_i, n) - x_i p. \tag{22}$$

Notice from Fig. 2(b) that the perturbation variance is fairly constant across different complexities for each sequence, which means slope parameter a_i is small. Even for sequences such as *Mobile*, the perturbation parameter a_i is about 10% the slope of its quality function. Moreover, the high correlation coefficient ρ further reduces the effect on the cost change between adjacent intervals, such that the perturbation-to-quality ratio is $\sqrt{\rho}\sigma_i(x_i)/Q_i(x_i)$. Hence, as long as the initial cost converges, the FCA cost function should be approximately identical to SWM-U, leading to near-identical resource allocations, as verified in the next section.

VI. Simulations and Results

To measure the performance of our resource management algorithms, we used the video coder from [18] 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 methodology can just as easily apply to any other quality-resource scalable video coding algorithms (e.g., encoding with different macroblock sizes

TABLE III
AVERAGE NUMBER OF CYCLES PER FRAME FOR TASK TO PE ALLOCATIONS AFTER FOUR ITERATIONS OF THE PA ALGORITHM

Task/Processor	PE 1	PE 2	PE 3	PE 4	PE 5
Foreman 1	267982	3564	4333	4333	4019
Foreman 2	3564	267982	4334	4333	4019
Coastguard 1	3237	3237	299839	3885	3571
Coastguard 2	3237	3237	3884	299842	3571
Mobile 1	3310	3310	4077	4078	289221

and prediction modes in H.264). In order to verify that our algorithms are suitable for DVS-enabled PEs, we used the StrongARM processor profile [26].

A. SWM/FCA Behavior for Dynamic Video Quality Functions

In Fig. 3, the iterations of the SWM for computational resource allocations are plotted for two *Foreman*, two *Coastguard*, and one *Mobile* sequences sharing the same system. Note that only one curve for the *Foreman* and *Coastguard* sequences are plotted, since resource allocations are identical for the same sequences. Negative indices refer to iterations for global adaptation (via the SWM-based algorithm), such that the allocations can converge before the video starts playing at iteration/time 0. The MEP iterations for the SWM-U and FCA are then synchronized with the video sequence during runtime, such that one iteration of the MEP is performed between each GOP. (The video quality function changes after each GOP.) Note that we purposely ran eight iterations of the SWM algorithm during global adaptation before running the FCA to allow the cost to converge. On the other hand, four iterations were sufficient for SWM global adaptation before running SWM-U, since the SWM-U continued to adapt its cost metric and resource allocations accurately during runtime under *very low complexity* (i.e., only a single MEP iteration is required for accurately updating the resource allocation between each GOP!). Moreover, because the cost fluctuates little upon convergence [Fig. 3(c) and (f)], the FCA, which requires no communications overhead except at the beginning of the video sequence, performs almost equally to the SWM-U. Importantly, the SWM-U required only an adaptation time of a single interval to be within 3% of the optimal resource allocation, while the FCA maintained a 5% deviation from the optimal resource allocation throughout.

B. EEM Convergence and Tradeoff Parameter λ

The convergence of the decentralized EEM algorithm to the optimal SWMEM solution is shown in Fig. 4, based on the StrongARM energy model [26], where the voltage V is proportional to the frequency f, and hence the energy $E \propto V^2 \propto f^2$. The same five video tasks above were also used for this experiment. The step size parameter is fixed at $\gamma = 1$, such that the real resource demand is projected for each application [see (8)], and the energy weight is set to

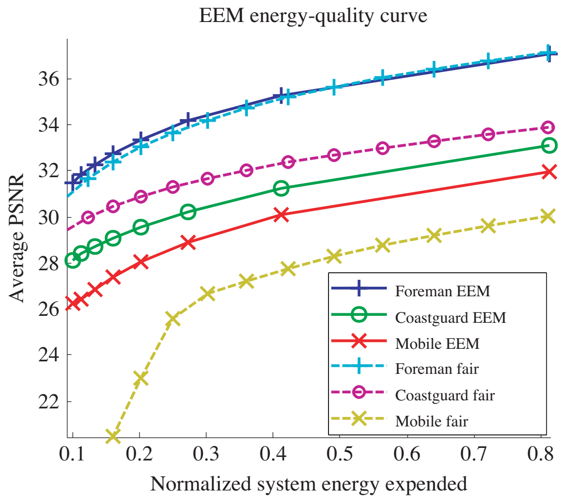
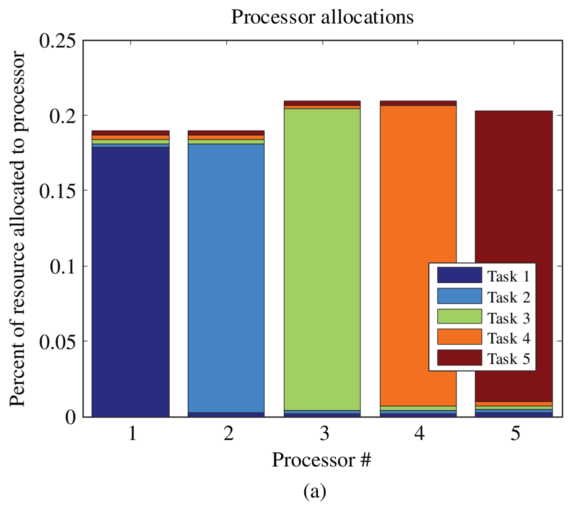


Fig. 6. Energy–quality curves for video sequences using Nash product fairness, compared against the fair share scheduling algorithm.

$\lambda = 2 \times 10^{-3}$PSNR/J. The EEM converges quickly (after only five iterations) and is therefore ideal for resource allocation in dynamic environments.

In Fig. 5, we plotted the video qualities and the energy consumed for different values of λ in the SWMEM/EEM, and compared it against a complexity fair scheduling scheme. 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 Fig. 5b, is 5–10 dB higher for the EEM than for fair share resource allocation at low-power regions, which translates to a significantly higher average video quality per task (1–2 dB PSNR). We also compared the complexity-fair scheduling scheme against the Nash product social welfare objective (a utility fair scheme), given by the product of application utilities (Fig. 6) [3]. This can be achieved by requiring each task to use a utility function that is the log of its quality function, i.e., $\ln(Q_i(x_i))$. Note that the Nash product-based EEM algorithm is a fairer allocation of resources, since the quality of the *Mobile* sequence, which requires far more computational resources than the *Foreman* and *Coastguard* sequences, is greatly increased.



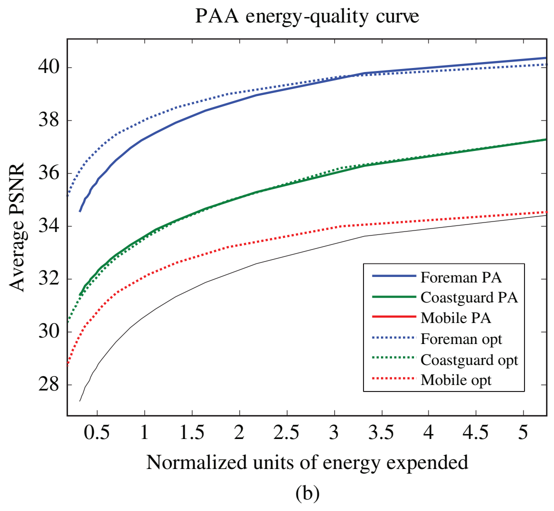


Fig. 7. (a) Bar graph of the resulting processor resource distribution after four iterations of the PA algorithm. Tasks 1, 2 are *Foreman*, 3, 4 are *Coastguard*, and 5 is *Mobile*. (b) The energy–quality curves for the PA algorithm and the optimal SWMEM solution.

TABLE IV
COMPARISON OF SWMEM-S AND RATE MONOTONIC (NORMALIZED) COMPLEXITY ALLOCATION FOR VARIOUS TIME INTERVALS, AND RESULTING (NORMALIZED) ENERGY AND QUALITIES UNDER FIXED ENERGY CONSUMPTION. THE SAME SETS OF TASKS WERE USED AS IN THE PREVIOUS SUBSECTIONS

	Rate monotonic			SWMEM-S/EEM-S		
Time interval/priority	1	2	3	1	2	3
Task 1	75.0	60.0	120.4	108.2	74.0	43.9
Task 2	75.0	60.0	120.4	108.2	74.0	43.9
Task 3	200.0	190.2	154.8	201.6	161.1	128.7
Task 4	200.0	190.2	154.8	201.6	161.1	128.7
Task 5	200.0	249.7	199.7	199.3	160.5	129.0
Total complexity	750.0	750.0	750.0	818.9	630.7	474.1
Total energy	12.66E			9.07E		
Average PSNR	29.75 dB			29.80 dB		

C. Allocation Behavior for the Processor Assigning Algorithm

We simulated the performance of the PA algorithm by assigning the five video decoding tasks to five separate, identical DVS-enabled processors. Note that from Table III and Fig. 7a, for five tasks and five processors, the final resource distribution allocates most of each task's complexity to its assigned processor after only four iterations of the PA algorithm, which demonstrates that using tax functions can compel tasks to run mostly on their assigned processors.

We also performed simulations to compare the PA algorithm with the ideal SWMEM solution when tasks can be arbitrarily parallelized across all PEs (See Fig. 7b). Note that for very small λ, or high energy availability regions, the processor dedicating algorithm achieves approximately the same performance as the SWMEM. This may be explained by the fact that when energy is cheap, the video qualities achieved are near maximum and increase very little per unit energy allocated. Consequently, resource allocation is not an important issue when resources are highly available. However, the PA algorithm yields considerably less quality than the ideal SWMEM solution when the cost per unit energy λ is high, since the slope of video quality functions are much steeper and differ much more between sequences at low energy/quality levels. Indeed, the poor performance of the PA algorithm is largely a result of poor PE-to-task assignment, which indicates that smartly assigning/sharing processors is very important when energy is scarce.

D. Joint SWMEM and Scheduling Simulation

The SWMEM-S simulations were performed for a three-level hierarchy of frames, and the quality-resource curves were used for the different bitplane-truncation points for each frame. Note that if a base layer frame is decoded down to a certain bitplane, decoding the second and third layers to a finer quantization level does not contribute much to the overall quality, since the distortion created by the base layer frame still exists. Hence, the quality of the decoded frames are interdependent. As shown in Table IV, by decoding the different priority frames during different time intervals and adapting the processor frequency in every interval based on application job dependencies, we can achieve better performance than rate-monotonic DVS, which is the optimal application-agnostic policy for saving energy given hard deadlines. Using the same five video decoding tasks as above, we showed that the SWMEM-S achieved approximately the same video quality as rate-monotonic DVS while providing over 28% in energy savings.

VII. Conclusion

In this paper, we presented a low-complexity informationally decentralized resource management solution for multiple multimedia applications sharing a resource-constrained system. We verified that our solution can be used to optimally achieve various objectives, such as maximizing the social welfare of applications, minimizing system energy consumption, assigning processors to applications, and efficiently updating the resource allocation in dynamic environments. Importantly, these solutions can be achieved even when applications are unwilling to reveal private information about their utilities. An avenue for future work is to construct and evaluate informationally decentralized algorithms based on more sophisticated (and possibly non-convex) resource models that deal explicitly with various elements and features that can exist within a heterogeneous system (e.g., shared memory and caches between PEs, special-function processors, voltage islands, etc.).

Appendix

Proof of Proposition 1

Proof: To show that the decentralized task objective functions are at equilibrium at the optimal point of the SWMEM objective function (6), and that the equilibrium point is unique, we use the KKT conditions for optimality. Since the SWMEM function is concave, the KKT conditions for optimality exist at a unique point, where for some $\vec{\mu} = (\mu_1, \mu_2, \ldots, \mu_I) \geq 0$, the following conditions hold:

$$\nabla\left(\sum_{i=1}^{I} Q_i(x_i) - E^*\left(\sum_{i=1}^{I} x_i\right)\right) - \mu = 0 \qquad (23)$$
$$\mu_i r^{(i)} = 0, \quad i = 1, \ldots, I.$$

The KKT conditions for the task-level optimization using the EEM tax function are

$$\frac{d}{dx_i} Q_i(x_i) - \frac{d}{dx_i} E^*\left(\sum_{i=1}^{I} (x_i + d_i)\right) - \mu_i = 0 \qquad (24)$$
$$\mu_i x_i = 0.$$

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 (8), the excess demand $d_i = \sum_{l \neq i} x_l$. It can be seen that at equilibrium, the equation involving the ith component of the gradient in (23) is identical to the expression in (24), i.e.,

$$\begin{aligned}\nabla_i&\left(\sum_{i=1}^{I} Q_i(x_i) - E^*\left(\sum_{i=1}^{I} x_i\right)\right) - \mu_i \\ &= \frac{\partial}{\partial x_i} Q_i(x_i) - \frac{\partial}{\partial x_i} E^*\left(\sum_{i=1}^{I} x_i\right) - \mu_i \\ &= \frac{d}{dx_i} Q_i(x_i) - \frac{d}{dx_i} E^*(x_i + d_i) - \mu_i. \qquad (25)\end{aligned}$$

Since the intersection of all KKT conditions for all task-level objective functions to be in equilibrium is precisely the condition for system global optimality, convergence of the decentralized algorithm (EEM) guarantees an efficient solution to the global objective function (SWMEM).

References

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

[2] D. Garlan, D. Siewiorek, A. Smailagic, and P. Steenkiste, "Project aura: Toward distraction-free pervasive computing," *IEEE Pervasive Comput.*, vol. 21, no. 2, pp. 22–31, Apr.–Jun. 2002.

[3] Y. Chevaleyre, P. E. Dunne, U. Endriss, J. Lang, M. Lematre, N. Maudete, J. Padget, S. Phelps, J. A. Rodrguez-Aguilar, and P. Sousa. *Issues in multiagent resource allocation* [Online]. Available: http://www.doc.ic.ac.uk/.ue/MARA/mara-may-2005.pdf

[4] V. Poladian, J. P. Sousa, D. Garlan, and M. Shaw, "Dynamic configuration of resource-aware services," in *Proc. 26th ICSE*, 2004, pp. 604–613.

[5] V. Poladian, J. Sousa, F. Padberg, and M. Shaw, "Anticipatory configuration of resource-aware applications," *ACM SIGSOFT Software Eng. Notes*, vol. 30, no. 4, pp. 1–4, Jul. 2005.

[6] R. Rajkumar, C. Lee, J. Lehoczky, and D. Siewiorek, "A resource allocation model for QoS management," in *Proc. IEEE Real-Time Syst. Symp.*, San Francisco, CA, 1997, pp. 298–307.

[7] S. Ghosh, R. R. Rajkumar, J. Hansen, and J. Lehoczky, "Scalable resource allocation for multi-processor QoS optimization," in *Proc. 23rd IEEE Int. Conf. Distributed Comput. Syst. (ICDCS '03)*, May 2003, pp. 174–183.

[8] L. Capra, W. Emmerich, and C. Mascolo, "CARISMA: Contextaware reflective middleware system for mobile applications," *IEEE Trans. Software Eng.*, vol. 29, no. 10, pp. 929–945, Oct. 2003.

[9] M. Chiang, S. H. Low, A. R. Calderbank, and J. C. Doyle, "Layering as optimization decomposition: A mathematical theory of network architectures," *Proc. IEEE*, vol. 95, no. 1, pp. 255–312, Jan. 2006.

[10] J. Chakareski and P. Frossard, "Rate-distortion optimized distributed packet scheduling of multiple video streams over shared communication resources," *IEEE Trans. Multimedia*, vol. 8, no. 2, pp. 207–218, Apr. 2006.

[11] J. Huang, Z. Li, M. Chiang, and A. K. Katsaggelos, "Joint source adaptation and resource pricing for multi-user wireless video streaming," *IEEE Trans. Circuits Syst. Video Tech.*, vol. 18, no. 5, pp. 582–595, May 2008.

[12] *List of Video Codecs* [Online]. Available: http://www.fourcc.org/codecs.php

[13] L. Benini and G. De Micheli, *Dynamic Power Management: Design Techniques and CAD Tools*. Norwell, MA: Kluwer, 1997.

[14] W. Yuan, K. Nahrstedt, S. Adve, D. Jones, and R. 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.

[15] T. Stoenescu and J. Ledyard, "A pricing mechanism which implements a network rate allocation problem in nash equilibria," *IEEE/ACM Trans. Netw.*, to be published.

[16] X. Lin, N. Shroff, and R. Srikant, "A tutorial on cross-layer optimization in wireless networks," *IEEE JSAC*, vol. 24, no. 8, pp. 1452–1463, Aug. 2006.

[17] T. Ishihara and H. Yasuura, "Voltage scheduling problem for dynamically variablevoltage processors," in *Proc. ACM ISLPED*, 1998, pp. 197–202.

[18] B. Foo, Y. Andreopoulos, and M. van der Schaar, "Analytical complexity modeling of wavelet-based video coders," in *Proc. ICASSP 2007*, vol. 3. Honolulu, HI, pp. 789–792.

[19] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Univ. Press, 2004.

[20] N. Mastronarde and M. van der Schaar, "A bargaining theoretic approach to quality-fair system resource allocation for multiple decoding tasks," *IEEE Trans. Circuits Syst. Video Technol. (TCSVT)*, vol. 18, no. 4, pp. 453–466, Apr. 2008.

[21] E. Akyol and M. van der Schaar, "Complexity model based proactive dynamic voltage scaling for video decoding systems," *IEEE Trans. Multimedia*, vol. 9, no. 7, pp. 1475–1492, Nov. 2007.

[22] Z. Yu and J. Zhang, "Video deblocking with fine-grained scalable complexity for embedded mobile computing," in *Proc. 7th Int. Conf. Signal Process. 2004 (ICSP '04)*, Aug. 2004, pp. 1173–1178.

[23] 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.

[24] N. A. Lynch, *Distributed Algorithms*. San Mateo, CA: Morgan Kaufmann, 1996, pp. 51–80.

[25] D. Bertsekas, *Nonlinear Programming*. New York: Palgrave Macmillan, 1997.

[26] A. Sinha and A. P. Chandrakasan, "Jouletrack: A web based tool for software energy profiling," in *Proc. IEEE/ACM DAC*, 2001, pp. 220–225.

Brian Foo obtained the Ph.D. degree in 2008 in electrical engineering from the University of California, Los Angeles, under the supervision of Prof. Mihaela van der Schaar.

He is currently at the Advanced Technology Center, Lockheed Martin Space Systems Company, Sunnyvale, CA, as a Research Scientist.

Mihaela van der Schaar (SM'04) is an Associate Professor in the Department of Electrical Engineering, University of California, Los Angeles.

She holds 30 granted U.S. patents and three ISO awards. She received the NSF Career Award in 2004, the Best Paper Award from IEEE Transactions on Circuits and Systems for Video Technology in 2005, the Okawa Foundation Award in 2006, the IBM Faculty Award in 2005, 2007, and 2008, and the Most Cited Paper Award from *EURASIP: Image Communications journal*, in 2006.