Coordinating Energy-Aware Adaptation of Multimedia Applications and Hardware Resource

Wanghong Yuan, Klara Nahrstedt, Xiaohui Gu

Department of Computer Science
University of Illinois at Urban-Champaign
Urbana, IL 61801, USA
{wyuan1, klara, xgu}@cs.uiuc.edu

ABSTRACT

This paper presents a middleware framework to coordinate adaptation of multimedia applications and their underlying hardware resource in mobile devices, so that multimedia applications are able to provide user-preferable QoS requirements under resource contention and energy constraints. The major contributions of the framework are: (1) energy-aware processor reservation for multimedia applications, (2) formal models of hardware adaptability, application adaptability, and user preference, and (3) an integrated algorithm to coordinate adaptation of multimedia applications and hardware resource.

Keywords

Adaptation, Middleware, Quality of Service

1. INTRODUCTION

Multimedia applications are becoming increasingly an integral part of the ubiquitous computing environment using battery-powered mobile devices, where application quality of service (QoS) requirements and system resource constraints are both demanding and dynamic. Multimedia applications can usually adapt their behavior to trade off their QoS requirements for resource demands. Moreover, modern mobile devices are also able to switch among multiple operating modes, for example, multiple processor speeds, with tradeoff between performance and energy consumption. A challenging problem is how to coordinate the adaptation of both multimedia applications and hardware resource under energy constraints.

Resource reservation, adaptation, and optimization are generally used to support various QoS requirements of multimedia applications. However, most proposed approaches [2][3][6][7][8][12] have treated the underlying hardware architecture as a static entity, which provides a fixed amount of system resource. On the other hand, previous work on hardware adaptation, for example, dynamically adjusting processor speed [1][4][9][11], is oblivious to the applications, i.e., it provides no mechanism to notify software applications about the hardware resource status changes.

In this paper, we present a middleware framework to coordinate the adaptation of multimedia applications and their underlying hardware resource in an integrated fashion. The goals of the middleware framework are: (1) soft real-time guarantee of multimedia application deadlines, (2) sufficient energy for multimedia applications to finish their tasks, and (3) minimum wasted energy. Our framework makes three major contributions to meet the above goals. First, it allows multimedia applications to make energy-aware processor reservation, and makes admission control based on both processor utilization and energy availability. Second, it formally models hardware adaptability, application adaptability, and user preference. Third, it adapts processor and multimedia applications in a coordinated way, thereby delivering sufficient processor and energy resources to multimedia applications, and operating the processor as slow as possible to save energy.

The rest of the paper is organized as follows. First, section 2 introduces the system architecture. Section 3 describes the coordinating adaptation algorithm. Section 4 presents the simulation evaluation. Finally, section 5 provides some concluding remarks.

2. SYSTEM ARCHITECTURE

The system architecture consists of multimedia applications, hardware resource, and middleware between them, as shown in Figure 1. The application QoS profile describes the multiple QoS points of a multimedia application, each with a separate QoS requirement and resource cost. The user QoS profile describes user's preference of application QoS, which determines the adaptation order between QoS points. The hardware resource profile describes the multiple operating points of resource, each with a separate performance and energy consumption.

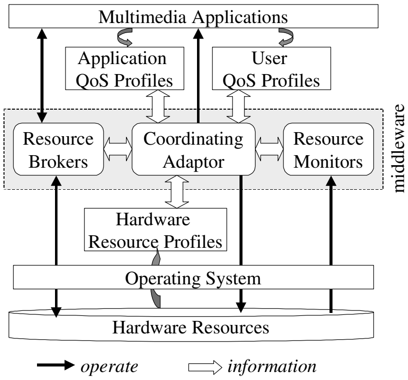

Figure 1. System Architecture

Multimedia applications make energy-aware resource reservation[1], requesting the resource corresponding to their current QoS points. Resource brokers make admission control on reservation request and allocate resource. Resource monitors observe resource status, such as processor utilization and battery lifetime. The coordinating adaptor adapts both multimedia applications and hardware resource, so that (1) all applications are allocated resource requested to meet their QoS requirements; and (2) hardware components save energy as much as possible.

3. ADAPTATION ALGORITHM

The coordinating adaptor adapts both hardware resource and multimedia applications based on processor utilization, energy constraint, and the QoS requirements and corresponding resource cost of applications. This section first describes the hardware resource profile, the application QoS profile, and the user QoS profile, and then presents the adaptation algorithm.

3.1 Hardware Resource Profile

Some modern processors, such as Mobile Intel Pentium III [5] and Crusoe TM5600 [10], support multiple processor speeds with tradeoff between performance and energy consumption. In our framework, each processor speed is referred to as an *operating point*, and all possible operating points together form the *operating space* of the processor. Each operating point is associated with a separate processor capacity, normalized to the number of millions of instructions per second (MIPS), and system power (i.e., the energy consumption rate).

System power is essentially proportional to the processor speed. The battery lifetime is inverse proportional to system power, and can be calculated according to the current energy and system power of an operating point. Therefore, the higher the speed is, the higher the processor capacity and system power are, and the shorter the battery lifetime is, as shown in Figure 2.

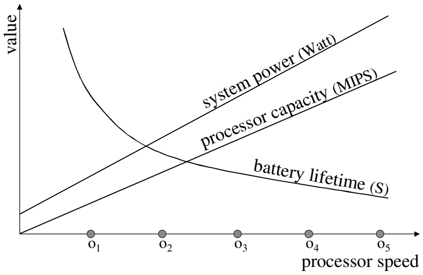

Figure 2. Processor Operating Space

The hardware resource profile describes the operating points in the operating space. Without loss of generality, operating points are ordered according to the processor capacity. We summarize above concepts as follows:

Operating Space: $O = \{o_1, o_2, \ldots, o_n\}$ (1)

Processor Capacity: $c: O \rightarrow N$ (2)

System Power: $p: O \rightarrow N$ (3)

Remaining Battery Lifetime: $t: O \rightarrow N$ (4)

Operating Point Order: $o_s < o_t$, if and only if $s < t$ and $c(o_s) < c(o_t)$ (5)

where N is the natural number set

3.2 Application QoS Profile

A multimedia application needs to meet user QoS requirements with multiple parameters, and each parameter represents a QoS dimension, which contains a finite set of discrete quality values. The combination of quality value in each dimension is referred to as a *QoS point* of the multimedia application, which is associated with a resource cost, normalized to the number of millions of instructions per second (MIPS). All possible QoS points together form the *QoS space* of the multimedia application. For example, a video on demand (VOD) application provides QoS requirements with dimensions of frame rate, frame size, and color depth, as shown in Figure 3.

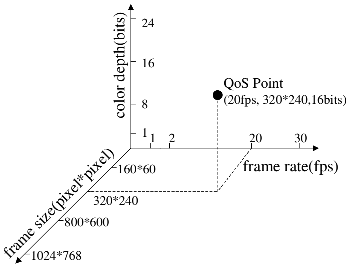

Figure 3. QoS Space of a VOD Application

The application QoS profile describes QoS points in the QoS space of a multimedia application, summarized as follows:

QoS Dimensions: $D_1, D_2, \ldots, D_m$, and $D_i = \{v_{i1}, v_{i2}, \ldots, v_{iS_i}\}$, where $1 \le i \le m$, and S_i is the number of values in the dimension D_i (6)

QoS Point: $q_j = (v_1, v_2, \ldots, v_m)$, where $v_i \in D_i$, and $1 \le i \le m$ (7)

QoS Space: $S = D_1 \times D_2 \times \ldots \times D_m$ (8)

Resource Cost: $r: S \rightarrow N$, where N is the natural number set (9)

3.3 User QoS Profile

From the user's point of view, different QoS dimensions may have different importance, for example, frame rate is more important than color depth in a VOD application. The user QoS profile describes the order of QoS points with user preference, and the QoS point order determines the adaptation path of application, as explained in next subsection. We assume that the more important a QoS point is, the higher the associated resource cost is, i.e.,

QoS Point Order: $q_j < q_k$ if and only if $r(q_j) < r(q_k)$ (10)

---

[1] In this paper, we focus only on processor resource, which, in turn, consumes energy resource. By energy-awareness, a multimedia application specifies its duration, which imposes an energy constraint, i.e. the battery should last until the application finishes.

3.4 Adaptation Algorithm

Multimedia applications make energy-aware processor reservation to meet their QoS requirements. *Processor utilization* of an application is the percentage of the requested processor resource, determined by the resource cost of its current QoS point and the processor capacity of the hardware operating point.

$$\text{Processor Utilization: } \theta\text{: } S \times O \rightarrow R,\ \theta(q,\ o) = r(q)/c(o), \text{ where } R \text{ is the real number set} \qquad (11)$$

The adaptation algorithm, as shown in Figure 4, maintains two system constraints: (*i*) R-EDF soft real-time scheduling [12] to multimedia applications, and (*ii*) enough remaining battery energy for all applications to finish their tasks, and then adjusts the processor to run as slow as possible to save energy. The adaptation algorithm is invoked when resource may need to be reallocated under events of reservation request, reservation release, or low energy availability. Without loss of generality, we assume that (*a*) the processor runs at operating point o_r; and (*b*) there are w multimedia applications, each with QoS point q_{ij} and remaining duration $d_i\ (1 \leq i \leq w)$, when the algorithm is invoked.

```
1. Event of reservation request with QoS point $p_{w+1}$
   IF $\sum_{i=1}^{w+1} \theta(q_{ij}, o_r) \leq 1 \,\&\&\, \max_{i=1}^{w+1} d_i \leq t(o_r)$,
      THEN admit without adaptation and return
   IF able to upgrade resource to operating point $o_s$ (s>r), so
      that $\sum_{i=1}^{w+1} \theta(q_{ij}, o_s) \leq 1 \,\&\&\, \max_{i=1}^{w+1} d_i \leq t(o_s)$
   THEN adapt resource to admit
   ELSE IF able to degrade applications to QoS point $q_{ik}$ ($q_{ik} < q_{ij}$ and $1 \leq i \leq w+1$), so that $\sum_{i=1}^{w+1} \theta(q_{ik}, o_s) \leq 1 \,\&\&\, \max_{i=1}^{w+1} d_i \leq t(o_s)$
       THEN adapt both resource and applications to admit
       ELSE reject
2. Event of reservation release with QoS point $p_w$
   IF able to first upgrade applications to QoS point $q_{ik}$ ($q_{ik} \geq q_{ij}$
      and $1 \leq i \leq w+1$), and/or degrade resource to operating point
      $o_s$ (s ≤ r), so that $\sum_{i=1}^{w-1} \theta(q_{ik}, o_s) \leq 1 \,\&\&\, \max_{i=1}^{w-1} d_i \leq t(o_s)$
   THEN adapt applications and/or resource
3. Event of low energy availability
   DO degrade applications QoS point $q_{ik}$ ($q_{ik} \leq q_{ij}$ and $1 \leq i \leq w+1$) and then degrade resource to operating point $o_s$ (s ≤ r)
   UNTIL $\sum_{i=1}^{w} \theta(q_{ik}, o_s) \leq 1 \,\&\&\, \max_{i=1}^{w} d_i \leq t(o_s)$
   IF successful
       THEN adapt resource and applications
       ELSE notify the user to take actions
```

Figure 4. Adaptation Algorithm

In the above algorithm, we employ two *application-privilege* policies. (1) *As-many-applications-as-possible*: for reservation request, it first tries to upgrade resource, and then adapts already admitted applications if resource adaptation only cannot meet constraints; (2) *As-high-QoS-as-possible*: for reservation release, it first tries to upgrade QoS points of applications, and then degrade resource operating point. Note that the user may define some priorities for applications, so different applications adapts with different weights. For example, when battery lifetime is insufficient, some applications may maintain their current QoS points, while others degrade a lot or exit.

4. SIMULATION RESULTS

In this section, we evaluate the success rate of resource reservation and energy saving achieved by the proposed coordinating adaptation algorithm. The results in this section are initial, and obtained by simulation Table 1 shows the operating space of hardware resource, with initial total energy 1500 Joules. Table 2 shows the QoS space of a *MPEG player* application, with initial duration 150 seconds.

Table 1. Operating Space of Hardware Resource

Operating Point	o_1	o_2	o_3
Processor Capacity (MIPS)	80	100	120
Energy Consumption (Watt)	3	4	5
Initial Total Energy (Joule)	1500		

Table 2. QoS Space of A MPEG Player

QoS Point	q_1	q_2
Resource Cost (MIPS)	40	50
Initial Duration (second)	150	

At the beginning, the system runs at operating point o_1, and we start a copy of the MPEG player, with resource reservation request for QoS point q_2, every 40 seconds.

Figure 5 shows the adaptation of the processor capacity (i.e., operating points) of the hardware resource. Figure 6 shows the adaptation of the resource cost (i.e., QoS points) of the second MPEG player, starting at time 40 and ending at time 190.

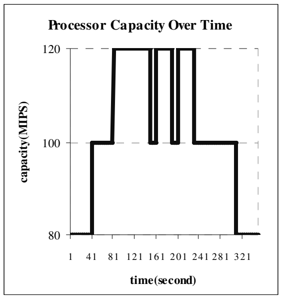

Figure 5. Adaptation of Processor Capacity

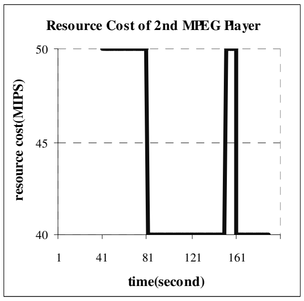

Figure 6. Adaptation of Application Resource Cost

Figure 7 shows the effectiveness of resource adaptation to save energy. We assume that the processor always runs at the highest speed (i.e., operating point o_3) without adaptation. The processor can only run for 300 seconds without adaptation. With resource adaptation, the processor tries to run as slow as possible, thereby consuming less energy, while meeting the resource demands of

MPEG players. It saves 210 Joules energy for the first 300 seconds, and the saved energy percentage is $210/1500 * 100\% = 14\%$. Figure 8 shows the effectiveness of application adaptation to admit more applications. We assume that the MPEG player always runs with the highest QoS requirement (i.e., QoS point q_2) without adaptation.

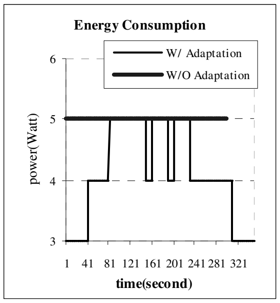

Figure 7. Comparison of Energy Consumption with and without Adaptation

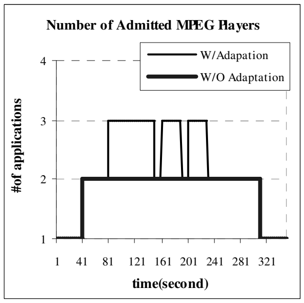

Figure 8. Comparison of the Number of MPEG Players with and without Adaptation

5. CONCLUSION

Multimedia applications are becoming increasingly an integral part of the ubiquitous computing environment using mobile devices, where applications and their underlying hardware resource adapt under energy constraints. In this paper, we have presented a middleware framework to coordinate the energy-aware adaptation of multimedia applications and hardware resource to meet two goals: (1) delivering enough processor and energy resource to multimedia applications, and (2) operating the processor as slow as possible to save energy.

This framework characterizes application QoS requirements, user preference, and resource operating modes with application QoS profile, user QoS profile, and hardware resource profile, respectively. Multimedia applications make energy-aware resource reservation, requesting the resource corresponding to their current QoS points. The coordinating adaptor first adapts the hardware resource to meet the application resource demands for their QoS requirements, and adapts multimedia applications to provide a different QoS requirement when system resource is overloaded or the battery energy is insufficient.

6. ACKNOWLEDGMENTS

This work was supported by the Air Force grant F30602-97-2-0121, the NSF CCR 96-23867, the NSF CISE CDA 96-24396, the NSF CISE EIA 99-72884, and the NASA grant under NASA NAG 2-1250. The views and conclusions contained in this paper are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of DARPA, NSF, NASA, or the U.S. government.

7. REFERENCES

[1] L. Benini, A. Bogliolo, and G. D. Micheli, A Survey of Design Techniques for System-Level Dynamic Power Management, *IEEE Transactions on VLSI Systems*, Vol.8, NO.3, 299-316, June 2000.

[2] H.H. Chu and K. Nahrstedt, CPU Service Classes for Multimedia Applications, *Proc. of IEEE Int. Conf. on Multimedia Computing and Systems*, 296-301, Florence, Italy, June 1999.

[3] J. Flinn and M. Satyanarayanan, Energy-Aware Adaptation for Mobile Application, *Proc. of the 17th ACM Symposium on Operating Systems Principles*, 48-63, Sout Carolina, December 1999.

[4] K. Govil, E. Chan, and H. Wasserman, Comparing Algorithms for Dynamic Speed-Setting of a Low-Power CPU, *Proc. ACM Int'l Conf. on Mobile Computing and Networking*, 13-25, Nov. 1995.

[5] Intel, Mobile Intel Pentium III Processor Featuring Intel SpeedStep Technology, [online] http://www.intel.com /design/mobile/perfbref/24956001.pdf, March 2001.

[6] M.B. Jones, D. Rosu, and M.-C. Rosu, CPU Reservations & Time Constraints: Efficient, Predictable Scheduling of Independent Activities, *Proc. of the 16th ACM Symposium on Operating Systems Principles*, Saint-Malo, France, 198-211, October 1997.

[7] C. Lee, J. Lehoczky, R. Rajkumar, and D. Siewiorek, On Quality of Service Optimization with Discrete QoS Options, *Proc. of the 5th IEEE Real-time Technology and Applications Symposium*, Vancouver, Canada, 276-286, June 1999.

[8] C.W. Mercer, S. Savage, and H. Tokuda, Processor Capacity Reserves: Operating System Support for Multimedia Applications, *Proc. of the Int. Conf. on Multimedia Computing and Systems*, 90-99, May 1994.

[9] T. Pering, T. Burd, and R. Brodersen, The Simulation and Evaluation of Dynamic Voltage Scaling Algorithms, *Proc. Int'l Symposium on Low Power Electronics and Design*, 76-81, August 1998.

[10] Transmeta, Crusoe Processor Model TM5600 Features, [online] http://www.transmeta.com/crusoe/download/pdf/ TM5600_ProductBrief_8-2-00.pdf, August 2000.

[11] M. Weiser, B. Welch, A. Demers, and S. Shenker, Scheduling for Reduced CPU Energy, *USENIX Symposium on Operating Systems Design and Implementation*, 13-23, Nov. 1994.

[12] W. Yuan, K. Nahrstedt, and K. Kim, R-EDF: A Reservation-Based EDF Scheduling Algorithm for Multiple Multimedia Task Classes, *Proc. of the 7th IEEE Real-Time Technology and Applications Symposium*, Taiwan, May 2001.