The Illinois GRACE Project: Global Resource Adaptation through CoopEration[*]

Sarita V. Adve[†], Albert F. Harris[†], Christopher J. Hughes[†], Douglas L. Jones[‡], Robin H. Kravets[†], Klara Nahrstedt[†], Daniel Grobe Sachs[‡], Ruchira Sasanka[†], Jayanth Srinivasan[†], Wanghong Yuan[†]
[†]Department of Computer Science
[‡]Department of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign
grace@cs.uiuc.edu

Abstract

Mobile systems primarily processing multimedia data are expected to become a dominant computing platform for a variety of application domains. The design of such systems imposes several new challenges, as it must consider demanding, dynamic, and multidimensional resource requirements and constraints, with energy becoming a first-class resource. At the same time, the ability of multimedia applications to trade off output quality for system resources and the difference between their peak and average demands offers a huge opportunity for optimization.

A promising approach to meeting the challenges of these systems, therefore, is to design all system layers with an ability to adapt in response to system or application changes. Further, to reap the full benefits of these adaptations, all system layers must cooperate to reach a system-wide globally optimal configuration. This paper describes the *Illinois GRACE – Global Resource Adaptation through CoopEration* – project, where our goal is to develop an integrated cross-layer adaptive system where hardware and all software layers cooperatively adapt to changing system resources and application demands, seeking to maximize user satisfaction while meeting resource constraints of energy, time, and bandwidth.

1 Introduction

Mobile devices employing wireless communication and primarily processing multimedia data (video, audio, and images) are expected to become a dominant computing platform. These systems offer new challenges and new opportunities compared to conventional desktop systems. New challenges arise because the target applications have demanding computational requirements that must be met in real time (e.g., video encoding for teleconferencing), with limited battery energy and an unreliable communication channel. The design of such systems must therefore consider demanding, dynamic, and multidimensional resource requirements and constraints, including energy, bandwidth, and time.

However, such systems also offer new opportunities. First, the real-time and dynamic nature of the applications often results in some computational slack. For example, it is not beneficial to complete the processing of a video frame before the next frame is available; therefore, depending on the computation needed for the current frame, the processor could be slowed to save energy. Second, multimedia applications often allow a tradeoff between output quality and resource usage. For example, depending on available bandwidth and energy, a video computation could be adjusted to improve or reduce the quality of the processed video. Thus, multimedia applications present to the system a range of possible resource requirements, with possibly different output qualities.

Based on the above observations, we believe that a key to meeting the challenges of our target systems is to design *all* system layers with an ability to *adapt* in response to system or application changes. Further, to reap the full benefits of these adaptations, all system layers must *cooperate* with each other to determine a system-wide globally optimal configuration. For our target systems, we anticipate that much, if not most, of the future gains from adaptivity will come from such joint adaptation across system layers. For example, for real-time video delivery, the simple choice of the compression technique entails a tradeoff between computation time, energy and number of bits (bandwidth). For a globally optimal solution, we must also jointly consider the choices for error correction for the lossy wireless network and protocols for congestion on the wired part of the network. The presence of multiple applications contending for the same resources and the ability of the processor to adapt its performance/energy tradeoff further reduce the possibility that any one layer can by itself determine a globally optimal solution.

The *Illinois GRACE – Global Resource Adaptation through CoopEration* – project is designing such an adaptive system. Figure 1 captures the vision of the project. Part (a) shows current systems with mostly fixed and isolated system layers. Part (b) shows our future vision where all parts of the system cooperatively adapt as a community, towards a globally optimal utilization of resources. The goal of our project, therefore, is

[*]This work is supported in part by the National Science Foundation under Grant No. EIA 99-72884, CCR-9988199, CCR-0096126, and EIA 007-2043. Sarita Adve is also supported by an Alfred P. Sloan Research Fellowship.

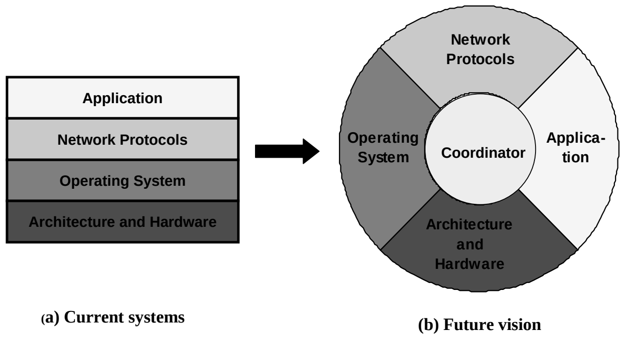

Figure 1: The GRACE approach – moving from fixed isolated layers to an adaptive global community.

to develop and demonstrate an integrated cross-layer adaptive system where hardware and all software layers cooperatively adapt to changing system resources and application demands, seeking to maximize user satisfaction while meeting resource constraints of energy, time, and bandwidth. Our project focuses on the hardware, network protocols, application, and resource management (scheduler) layers. This paper gives an overview of the GRACE project. It describes our proposed cross-layer adaptation framework, the support needed in the individual system layers for this framework, and some preliminary results.

There is a large amount of prior work on adaptation for all of the system layers we consider, but most focuses on adapting a single layer at a time, possibly in response to information from another layer (Section 2). Some work considers joint adaptation between two layers, but a single layer drives most of the adaptation (Section 2). For example, in [31], the application jointly controls the network layer error correction and its own video coding. There are several limitations of such an approach. The approach requires exposing the internals of one layer to another, mitigating the advantages of isolating different functionality within different layers. Software written in this manner is less reusable and the full flexibility of the system is less likely to be optimally exploited. Furthermore, it would be difficult to scale such an approach to joint adaptation between more than two layers. Using [31] as an example again, to exploit the flexibility in the network layer, an application must incorporate code to explicitly manage the network, each application must typically write its own version of this code (since it is tightly integrated with the application's work), and each application will make network layer decisions that are oblivious to other applications in the system and hence likely to be suboptimal. Further, scaling the approach in [31] to include the hardware layer would require the application to also manage adaptive hardware, exacerbating the above problems.

In contrast to previous work, we propose a system architecture that enables all system layers to cooperate in their adaptations, in a manner that greatly enhances software reusability, enables a globally optimal solution, and is scalable to multiple layers. In our framework, cooperation among system layers occurs through well-defined, minimal, and scalable interfaces. We allow enough coordination to seek a globally optimal solution, yet most of the work for adaptation occurs locally within each layer. This approach retains the advantages of previous fixed systems that isolate different functionality within different layers, and overcomes the limitations of the few previous approaches to joint adaptation.

More specifically, our framework limits inter-layer communication to two purposes. (1) A software layer queries other layers to get the cost of different configurations (e.g., querying hardware for energy cost). (2) All applications submit their possible configurations and costs to a resource manager, which acts as a mediator for choosing configurations to maximize system utility with available resources. Thus, we do not require a layer to know the internals of another layer, enhancing software reusability. The final decision of who gets which resources is made by a central resource manager, which has global information and hence is more likely to choose a globally optimal solution than any other individual layer. Finally, our solution scales to multiple layers because (i) much of the optimization process is local to each layer, and (ii) the central resource manager is limited to determining what resources (and consequent configurations) to assign an application and not how to determine these configurations.

Our project also differs from most previous work in seeking to jointly optimize only energy, time, and bandwidth (vs. only one or two resources). Finally, since we consider soft real-time multimedia applications, our constraints and opportunities differ from those for hard real-time and the more conventional non-real-time applications. Unlike the hard real-time work, our resource manager does not have to make worst-case assumptions and can give statistical guarantees. Unlike the non-real time work, we must consider that a “majority” of our requests need to be satisfied within the real-time specification, but can choose among multiple output qualities.

2 Prior Work

There is a tremendous amount of prior work on adaptation for all of the system layers we consider. For lack of space, we only mention some of the most closely related work and focus on how it differs from our approach.

A large amount of work has been done on hardware adaptation to save energy, using dynamic voltage (and frequency) scaling or DVS (e.g., [37]) and architectural adaptation (e.g., [26, 1, 9]). Most of this work, however, has not considered adaptive real-time applications, adaptive network protocols, bandwidth-driven adaptations, and/or interactions with the temporal scheduler.

Adaptive applications for mobile computing have focused on changes in available bandwidth [10, 36, 6, 28, 25] and energy [8, 7], as well as off-loading computation to fixed servers [27]. These solutions adapt or transform the data based on observing the state of the network or energy, but treat the network, hardware, and scheduling like a black box. This limits the success of such adaptations to the uncoordinated performance of the application and the underlying components of the system and hardware, which may provide services counter-productive to the goals of the application.

Protocols for mobile communications must adapt to existing conditions and take into account the effects of their adaptations on energy consumption [3, 41, 21] or timeliness [18, 13].

For this research, we focus on the issues of how much reliability to provide in a data stream and how to provide it, both with the consideration of energy costs. Solutions providing reliability through retransmission-based [29, 21] or forward error correction (FEC)-based techniques, as well as hybrid FEC/retransmission solutions [17, 34], generally do not consider the ability to adapt the level of reliability for the transmission stream [13, 20]. The limitation of all of these approaches is that pure network-based adaptation does not consider the requirements of specific applications and, therefore, has limited opportunities to improve the match of application requirements with available network resources. In addition, none of these approaches consider adaptive hardware or scheduling techniques.

There has been considerable exploration of joint adaptation between the application and network layers. Support for such adaptations has been provided by adaptive middleware [36, 19] or built directly into the application [5, 24, 33, 22]. However, this work has been limited by the control of the adaptations residing primarily within one layer, resulting in limitations discussed in Section 1. In particular, none of these previous works attempt to affect mutual adaptations at all system levels.

For resource management with multiple applications, prior relevant work includes resource allocation approaches for soft-real-time scheduling [12, 14, 23, 32], as well as QoS adaptation services in the resource management layer for assisting adaptive time-sensitive applications [38, 25, 2]. However, these techniques generally do not consider hardware adaptation or energy. Some recent work has considered energy scheduling with hardware adaptation in a limited way; e.g., [30] considers *hard* real-time embedded systems and so makes conservative worst-case assumptions about task processing times, and only considers DVS adaptation.

3 A Framework for Adaptation

The centerpiece of the GRACE project is a cross-layer adaptation framework that enables coordination of the adaptations of the different system layers for the best QoS possible. The key challenge lies in exposing only relevant information across layers without compromising the current advantages of having system layers that are virtually closed to each other (e.g., selective transparency). Thus, our solution must have the following properties. (1) It must perform its global cross-layer optimizations without exposing implementation internals of a layer to other layers. (2) It must localize adaptation decisions specific to a layer to within that layer. (3) A system component that exploits adaptation capabilities in other system layers must also be usable with implementations of those layers that do not have the same adaptation capabilities.

The above properties are the reason why we cannot simply take current solutions that consider joint optimizations between two layers and scale them to our problem. As explained earlier, that work involves a tight coupling between the two layers that are adapted, violating the above properties. Our proposed system, summarized in Figure 2, has the following key characteristics:

- *Individual application components adapt locally,* without knowledge of the internals of other parts of the system.
- Each software configuration is characterized by its *cost* (to represent its resource usage) and its *utility* (to represent user satisfaction). An application software component uses these metrics to drive its local adaptations (with the goal of minimizing cost for maximal utility).
- A software configuration determines its cost using *dynamic feedback* from the hardware and possibly other application components (e.g., the network).
- Since all resources are capable of adaptation, each resource offers *multiple operating points* and hence multiple possible costs (e.g., multiple combinations of execution time and energy) for a given software configuration.
- The resource manager receives requests from multiple applications with multiple associated costs and utilities. The resource manager selects the software and hardware configurations that will maximize overall system utility and meet the system constraints. Thus, the resource manager is not concerned with how an optimal configuration is reached within a layer, but is simply a mediator for ensuring that the selected configurations maximize overall system performance within the given constraints.
- Once the resource manager allocates a reservation, different system components are free to adapt locally without going through the resource manager, as long as they do not exceed the provided reservations.

Operationally, we envisage a system alternating between two major modes: *global* adaptation and *local* adaptation. Global adaptation involves the resource manager, where applications renegotiate for a new resource allocation. In local adaptation, the application and system layers may adapt to small and local changes in the system without going through the resource manager, as long as they do not exceed allocated resources or reduce overall utility. We expect the local adaptation mode to be the steady state mode.

A global adaptation is triggered by the resource manager under one of the following circumstances. (1) The resource usage of a current application changes; e.g., the input stream undergoes a significant change. (2) An application seeks admission into or leaves the system. (3) System characteristics change; e.g., the battery life goes below a certain threshold or the network becomes significantly more congested.

For purposes of discussion, assume the case of a new application seeking admission into the system. This application must specify its resource requirements (the requirements themselves may be determined by a combination of dynamic profiling via queries to the appropriate resource and programmer specification). If the resources required are available with the current hardware configuration, the resource manager will admit this application immediately. If the resources are not available, our proposed resource manager will first explore a hardware reconfiguration; e.g., increase the processor frequency to meet the application deadlines. If no hardware reconfiguration can meet

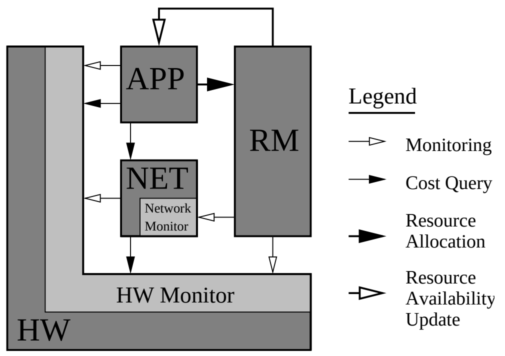

Figure 2: Flow of control in the GRACE cross-layer adaptation framework (only requests are shown). App is an application component; NET is the network software layer that runs on behalf of the application; RM is the resource manager; HW is hardware.

the requirements of the new application, then the resource manager will consider software reconfiguration as follows.

If the relative utility of running the new application is higher than that of maintaining the current level of service for applications already running, the resource manager will request the current applications to run at a lower cost operating point with potentially lower utility. This triggers a phase where each application explores multiple configurations (at possibly different utilities) using feedback from the hardware and other associated software layers (e.g., network protocols) to determine the cost (e.g., execution time, energy, bandwidth) of the configuration and guide the optimization. This process is further complicated by adaptations possible in the hardware and other software layers. As a result, these layers report multiple possible costs rather than a single cost for a given application configuration. On determining the local optima for reasonable utility values, the application submits a request to the resource manager containing the proposed utilities and associated costs (including the hardware and network protocol configurations associated with those costs). The resource manager examines the request in the context of the entire system and picks the appropriate configuration for the application, hardware, and network protocols that will maximize overall system utility and meet the resource constraints.

The above process repeats when any of the triggers for adaptation given above are satisfied. The global adaptation process can be viewed as a negotiation between different applications for resources, and the resource manager acts as an intermediary to ensure that all applications are treated fairly and do not exceed the available system resources.

Once the resource manager allocates resource reservations, different system layers are free to adapt locally (e.g., in response to local variations in the application input streams) as long as they do not exceed the provided reservation. If these local adaptations produce consistently lower execution time, energy, or bandwidth relative to the reservation, the resource manager will trigger a new global adaptation phase to change the reservation as described above.

Figure 2 summarizes the flow of control with our framework (only requests are shown for simplicity). There are four types of flows. An application and its associated components make cost queries (small solid arrows) to the available resources (hardware and network) to make a cost estimate. The application uses this estimate to reserve resources with the resource manager (large solid arrow). Implicit in this reservation is the configuration of the hardware and network chosen for the application and its components. During execution, the application and its components and the resource manager monitor current resource usage (small hollow arrows) to determine local optimizations. The resource manager also uses this information to track resource usage by a specific application, and notifies the application of any resource violation by it or any of its components (large hollow arrow).

Subsequent sections elaborate on the support needed within each system layer to implement the above framework.

4 Hardware Layer

To support our cross-layer adaptation framework in the hardware layer, we need to determine (1) useful hardware adaptations, (2) control algorithms for local adaptation, and (3) efficient techniques to predict cost for (and monitor) given software. Our work so far has mostly focused on energy-related adaptations and on the processor subsystem [15, 16, 35].

Our recent work in [16, 35] has focused on jointly employing DVS and architecture adaptation, and performing this joint adaptation both at the granularity of a frame and within a frame of a multimedia application. The architecture adaptations considered include changing the instruction window size of an out-of-order processor and changing the number of active functional units and resultant issue width. The work described below assumes static software and application deadlines assigned by the resource manager.

In [16], we developed a control algorithm for DVS and the above architecture adaptations, triggered at the granularity of a frame of a multimedia application. For each frame, this *inter-frame* algorithm picks the lowest energy configuration (i.e., frequency/voltage and architecture) that will meet the application deadline. Thus, the algorithm aims to slow down the processor as much as possible without violating the application deadline and using the lowest energy combination of architecture and frequency/voltage. The algorithm may choose different configurations for different frames, since they may have different amounts of computation.

The algorithm requires an efficient technique to predict the time and energy that the next application frame will take on each possible hardware configuration. We have developed such a technique based on empirical characterizations of multimedia application behavior [15] (e.g., the instructions-per-cycle, or IPC, for an architecture configuration stays roughly the same for frames of the same type for an application). These characterizations enable the use of simple profile information and a simple instruction count predictor to determine execution time and energy of the next frame for all hardware configurations. Since the adaptations are invoked at the granularity of a frame, overheads from profiling, prediction, and invoking an adaptation are relatively small.

We have also developed algorithms to perform architectural adaptations within an application frame (*intra-frame algorithms*), which exploit the variability in the execution profile within a frame [35]. These algorithms attempt to save energy without slowing down the processor. Since these adaptations work at a finer granularity, only very low overhead techniques can be used (e.g., we do not use intra-frame DVS). The inter-frame and intra-frame adaptation hardware algorithms work in a manner analogous to the global and local adaptations of our overall cross-layer algorithm – the inter-frame algorithm picks the maximal configuration for a frame and the intra-frame algorithm adapts within that configuration.

We have integrated the inter-frame and intra-frame adaptation algorithms, and evaluated them with DVS using simulation [16, 35]. Overall, we find that all the techniques are effective and the best overall energy savings come from using them together. For the systems and applications studied, the combination of DVS and both forms of architecture adaptation gives an average energy savings of 82% over a non-adaptive architecture, 28% over a system with DVS and no architecture adaptation, and 66% over a system with architecture adaptation and no DVS [35].

Later sections describe some results with combining application adaptation with the above hardware adaptations. In the future, we will broaden our architecture work to a more comprehensive set of adaptations, within the processor and the rest of the system, and to include network bandwidth considerations. We will need to extend the control algorithms for performing these adaptations locally within the budget provided by the resource manager. This will require extending our methods to predict the cost (in terms of time, energy, and network bandwidth) of running different software configurations with these adaptations. Some adaptations may be best exercised as global adaptations through the resource manager (e.g., due to its high overhead as with DVS or impact on other concurrent applications), while others are best exercised as local adaptations. We need to make these assessments using well-defined models of overheads and benefits. To coordinate with the rest of the system, we need to develop a software layer that can be accessed by the rest of the system to monitor the hardware and determine the cost of different software configurations. Our prediction and profiling techniques developed for local adaptations will provide the basis for these predictions for the higher level software as well. The monitoring software layer will essentially act as an intermediary between the adaptive software and hardware, either directly or through the resource manager.

5 Application Layer

Our research in application adaptivity has two major attributes. First, it must provide for global system adaptation by presenting a set of operating points, each with an associated resource cost and utility. Second, once an operating point has been selected by the resource manager, the application must locally adapt to stay within its assigned resource and utility bounds as conditions vary around this operating point.

Conceptually, the application's optimizer presents to the resource manager a list of possible operational points, each describing the utility (in terms of concrete measures like mean-squared error, frame rate and size, or error rate) and resource (CPU, network, energy) requirements of a possible configuration. Resource requirements can typically be estimated through profiling and by querying the various resource monitors for resource usage as the application executes normally. These operational points are provided to the global optimizer by every application in the system, which chooses which resources to allocate to each one so as to maximize the overall utility of the system. The operational points are then returned to the application as a resource allocation, and corresponding acceptable range of utility expected of the application.

Once an operating point has been chosen, it is the responsibility of the application's optimizer to configure the application so as to achieve the highest attainable utility within the allocated resource bounds. This can be done quickly using a constrained gradient-descent optimizer for the utility function over the allocated resource space. Although this type of optimizer may not be optimal if the utility function is not convex, such optimizers are fast and have been shown to perform well for this type of application [11]. If the application finds that due to changing conditions it cannot achieve the agreed-on utility using its allocated resources, it reports this to the resource manager and triggers a global adaptation. This is done even if conditions have improved and the application is exceeding its specified utility; in this case, excess resources may have been allocated to the application, which could be more profitably used by other applications. Using this combination of local and global adaptations, we can achieve the benefits of global optimization without requiring constant reallocation of resources between applications.

We are currently examining a wireless video delivery application, and focusing on energy expended in the CPU and in network transmission. For our initial exploration, we are considering a local application adaptation using a dynamic adjustment of an escape threshold for motion search. This adjustment affects both the CPU energy (by changing the computation complexity) and the transmission energy (by changing the number of bits transferred), but does not affect the output quality. Our adaptive application dynamically monitors energy usage for both CPU and network transmission for each frame, and adjusts the escape threshold using a simple gradient-based method to minimize energy. After every frame is transmitted, the optimizer adjusts the escape threshold by a small amount; if the sum of computational and transmit energy for the current frame is greater than that of the previous frame, the direction of adjustment is reversed. Thus, the optimizer will tend to follow a gradient to a locally optimal value for the escape threshold.

We simulated the above application, with and without the adaptation, on the adaptive hardware described in Section 4. In one representative case, overall energy consumed was reduced by 6% even with this relatively simple application adaptation that affects only a small portion of the application's total runtime. Other simulations have shown that the combination of adaptive hardware and software provides synergistic reductions in total energy consumption. This is because, when combined with dynamic voltage scaling and architectural adaptation as discussed in Section 4, the effects of small differences in computational complexity are magnified.

In the future, we plan to explore a range of application adaptations that will require global coordination. We will also develop general interfaces between the application and other system layers, including a specification of utility and costs.

6 Network Layer

The key challenges in the network layer lie in determining useful protocol configurations, providing a powerful yet general interface supporting adaptive multimedia applications, assessing cost, and making local optimizations.

In a dynamic environment, the network layer must be able to respond to changes in the communication channel over which it is transmitting data and adapt protocol parameters to affect the reliability and timeliness of data transmission. These adaptation decisions in turn affect the energy consumption of the communication and its associated computation. Given the lossy nature of wireless communication, there is a direct tradeoff between the cost of communication (in terms of energy consumption) and the quality of the data stream. To compensate for such lossy environments, communication protocols can use retransmission-based reliability or Forward Error Correction (FEC). In previous work, we have shown that aggressive retransmission-based reliability mechanisms can improve the overall energy consumption of a mobile device [21]. Such an approach can be enhanced by integrating some level of FEC-based error correction based on channel quality. Our goal is an energy-efficient error correction framework that addresses both bit-level and packet level error correction in a hybrid FEC-based and retransmission-based solution.

Such a network-centric approach is limited to providing fixed levels of service. Building on our previous research [20], we will allow the application to define different levels of reliability for data with different requirements. We believe that such an approach allows us to address specific application layer reliability requirements with respect to the needs of application layer frames [4]. We consider various ARQ solutions, including the consideration of statistical reliability guarantees, and various FEC solutions, including progressive encoding techniques as well as inter- and intra-packet FEC. Adaptation will be based on information about application level frame size, path MTU (Maximum Transmission Unit), and the error characteristics of the channel.

The interface between the network and the application hides these network- and system-specific details from the application. The application only supplies information about its requested QoS (i.e., bandwidth and loss tolerance) in terms of parameters meaningful to it. More specifically, the application may request the creation of one or more data streams with parameters that may include a statistical model of the data to be delivered (how many and what size packets), the (possibly dynamic) deadlines for each packet, and what kinds of losses are permitted (e.g., losses of complete packets, delivery of packets that may have one or more bit errors, or missed deadlines).

Based on this information from the application, the network responds with a cost estimate for energy and time. The energy estimate includes energy for any protocol processing and data encoding as well as the physical transmission of the data across the channel. The network layer optimizes the energy and time costs by adapting protocol parameters in response to the application requirements (loss and bandwidth), the current state of the network (e.g., loss rate, loss burst size), and the adaptation capabilities of the hardware.

We estimate costs based on calibration and profiling. Calibration provides base-line measurements of energy consumption and raw bandwidth for each device. Initial energy consumption calibration can be performed off-line, and later integrated into the system allowing run-time monitoring and updating of calibration information. To determine the energy consumption of specific actions, a profile is created to outline the expected use of bandwidth for given communication configurations. This profile contains information such as reliability levels and execution times for specific bandwidth usage. To estimate energy and time of our protocol codes, we use the same interface to the hardware as the application layer. Calibration information is combined with these run-time profiles of a protocol to estimate the amount of energy being or expected to be consumed for specific actions.

7 Resource Manager

Our resource manager layer requires the following research: (1) determination of an optimal resource allocation and system configuration during global adaptation, (2) monitoring and reacting to system changes in invoking local or global adaptation, and (3) consideration of resource allocation for an application consisting of a group of dependent tasks.

Our work has so far concentrated on the design of the resource manager components, its implementation as a middleware software, and enhancement of one of the entities in the resource manager, the soft-real-time scheduler with DVS to achieve adaptive scheduling of multimedia tasks [39, 40]. The resource manager design currently aims to satisfy the following goals: (1) provide soft real-time scheduling to multimedia applications under different processor speeds and energy levels, (2) minimize wasted energy, and (3) differentiate applications with different performance requirements under low energy availability.

The resource manager consists of three major components: the Dynamic Soft Real-Time (DSRT) processor scheduler, the energy manager, and the coordinating Processor and Energy Resource Management (PERM). The DSRT scheduler allows multimedia applications to reserve processor resource and corresponding energy resource and monitors the system workload. The energy manager polls the hardware monitor for energy availability (i.e., the remaining battery life) and the processor energy consumption. The coordinating PERM framework determines the policies for how to adjust reservations according to energy availability, uses corresponding policies to differentiate applications in case of low energy availability, adjusts the processor speed to minimize wasted energy, and notifies adaptive applications if it cannot extend the battery life and meet the processor resource requirements of applications under low energy availability.

Specifically, our resource manager allows multimedia applications to make an energy-aware processor resource reservation, and makes admission control based on both processor utilization and energy availability. Furthermore, it dynamically adjusts the speed and energy consumption of the processor, when the system workload changes or the energy availability is not sufficient. Finally, it distinguishes soft real-time adaptive multimedia applications from best-effort applications when adjusting the processor speed, so that multimedia applications maintain their performance and best-effort applications tolerate more performance degradation.

To achieve, especially, the dynamic adjustment of speed and energy consumption of the processor within our Dynamic Soft Real-Time (DSRT) Scheduling management, we have integrated dynamic voltage scaling (DVS) into the soft-real-time (SRT) scheduling. This integration achieves energy saving of DVS while preserving performance guarantees of SRT scheduling. The integration of DSRT with DVS, called SRT-DVS, resulted in the design and deployment of a variable speed constant bandwidth server (VS-CBS) algorithm in DSRT. The VS-CBS server enables multimedia applications to reserve resource based on their average resource usage, and ensures the correctness of reservation admission and enforcement, thereby delivering resource guarantees to applications, in a variable-speed context. The goal is to keep multimedia applications at the same QoS point through their reservation as long as possible, only changing QoS points if the current QoS cannot be achieved through local or hardware adaptation.

In particular, we have proposed three SRT-DVS algorithms to reduce processor energy. The first algorithm integrated the DSRT with DVS without any CPU bandwidth reclamation. The second algorithm takes SRT-DVS and expands it with the bandwidth reclamation property. This algorithm reclaims the unused bandwidth if a job finished earlier than planned, ensuring the processor does not waste energy through idle cycles. The third adaptive SRT-DVS algorithm adapts the bandwidth allocation of best effort VS-CBS by comparing its ready queue length in terms of number of jobs with a threshold pair.

We have run simulations to compare SRT-DVS with no-DVS DSRT algorithms. These results, shown in Figure 3, confirm that all three SRT-DVS algorithms perform better than the DSRT scheduler without DVS in energy consumption. Comparing the three SRT-DVS algorithms, the adaptive SRT-DVS performs best in terms of energy savings. These results are true for various workloads (intensive SRT arrival, intensive BE arrival, sparse SRT arrival, sparse BE arrival). In particular, the three SRT-DVS algorithms achieve more energy saving when the BE and SRT arrivals are both sparse, because they can slow down the processor more due to the light workload.

Our future work will include research on global and local adaptation concepts and algorithms, and their adaptive models to provide in the resource manager a stable coordination among multimedia applications, network, and hardware. Furthermore, as adaptive applications will change and run in a coordinated manner with network tasks, we need to develop the concept of group management in the resource manager as one of the important mechanisms for coordination.

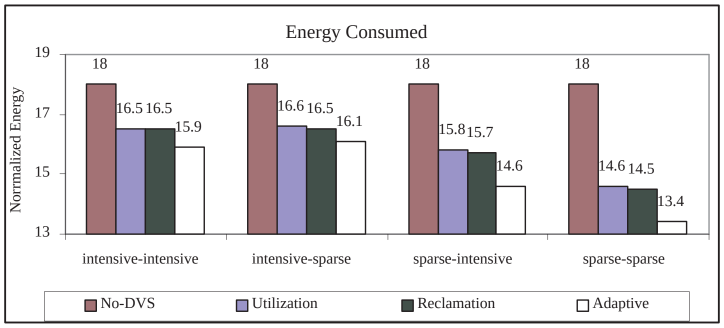

Figure 3: Comparison of SRT-DVS algorithms with no-DVS SRT scheduler, running MPEG player (25fps, 320x240pixels)

8 Conclusions

This paper describes the Illinois GRACE project, which proposes a novel approach for meeting the challenging demands of an increasingly dominant computing platform – mobile systems employing wireless communication and running multimedia applications. These systems will have demanding, dynamic, and multidimensional resource constraints, with energy as a first-class resource. At the same time, the ability of multimedia applications to trade off output quality for system resources and the difference between their peak and average demands offer a huge opportunity.

We have proposed a system architecture where all layers (hardware and software) are flexible and adapt cooperatively to best meet the real-time demands of the applications, within the available resources. For the systems we target, we believe that such joint cross-layer adaptation will be critical to achieving future benefits from adaptation. In contrast to the limited previous work on joint adaptation, we believe that our framework will enable cross-layer cooperation in a manner that greatly enhances software reusability, enables a globally optimal solution, and is scalable to multiple layers. At the same time, our approach retains the advantages of previous fixed systems that isolate functionality within different layers. We are currently developing support for such a framework within the hardware, network, application, and resource management layers, including simple but flexible APIs for these layers. Although we have made significant progress, much research is still required across the system layers to create a working system out of the above vision.

References

[1] D. H. Albonesi. Selective Cache Ways: On-Demand Cache Resource Allocation. In *Proc. of the 32nd Intl. Symp. on Microarchitecture*, 1999.

[2] V. Bharghavan et al. The timely adaptive resource management architecture. *IEEE Personal Communications*, 1998.

[3] I. Chlamtac, C. Petrioli, and J. Redi. Energy conservation in access protocols for mobile computing and communication. *Microprocessors and Microsystems Journal*, 1998.

[4] D. D. Clark and D. L. Tennenhouse. Architectural considerations for a new generation of protocols. In *Proceedings of the SIGCOMM '90 Symposium*, 1990.

[5] G. M. Davis and J. M. Danskin. Joint source-channel coding for image transmission over lossy packet networks. In *SPIE Conference on Wavelet Applications of Digital Image Processing XIX*, 1996.

[6] E. de Lara, D. S. Wallach, and W. Zwaenepoel. Puppeteer: Component-based adaptation for mobile computing, 2000.

[7] J. Flinn et al. Reducing the energy usage of office applications. IFIP/ACM International Conference on Distributed Systems Platforms 2001.

[8] J. Flinn and M. Satyanarayanan. Energy-aware adaptation for mobile applications. *SOSP*, 1999.

[9] D. Folegnani and A. González. Energy-Efficient Issue Logic. In *Proc. of the 28th Intl. Symp. on Comp. Architecture*, 2001.

[10] A. Fox et al. Adapting to network and client variation using infrastructural proxies: Lessons and perspectives. *IEEE Personal Communications*, 1998.

[11] M. Goel et al. A low-power multimedia communication system for indoor wireless applications. *Journal of VLSI Signal Processing Systems*, 2001.

[12] P. Goyal, X. Guo, and H. Vin. A Hierarchical CPU Scheduler for Multimedia Operating Systems. In *ACM Multimedia*, 1996.

[13] R. Han and D. Messerschmitt. A progressively reliable transport protocol. *Multimedia Systems*, 1999.

[14] H. hua Chu and K. Nahrstedt. Cpu service classes for multimedia applications. In *IEEE Multimedia Systems and Expo*, pages 296–301, Florence, Italy, June 1999.

[15] C. J. Hughes et al. Variability in the Execution of Multimedia Applications and Implications for Architecture. In *Proc. of the 28th Annual Intl. Symp. on Comp. Architecture*, 2001.

[16] C. J. Hughes, J. Srinivasan, and S. V. Adve. Saving Energy with Architectural and Frequency Adaptations for Multimedia Applications. In *Proc. of the 34th Intl. Symp. on Microarchitecture*, 2001.

[17] P. Karn. Toward new link layer protocols, 1994.

[18] B. Kim, Z. Xiong, and W. Pearlman. Progressive video coding for noisy channels. In *Proceedings ICIP 98*, 1998.

[19] R. Kravets, K. Calvert, and K. Schwan. Payoff adaptation of communication for distributed interactive applications. *Journal on High Speed Networking: Special Issue on Multimedia Communications*, 1998.

[20] R. Kravets et al. Adaptive variation of reliability. In *7th IFIP Conference on High Performance Networking*, 1997.

[21] R. Kravets, K. Schwan, and K. L. Calvert. Power-aware communication for mobile computers. In *6th Intl. Workshop on Mobile Multimedia Communications*, 1999.

[22] T.-H. Lan and A. H. Tewfik. Power optimized mode selection for h.263 video coding and wireless communications. In *Proceedings of ICIP*, 1998.

[23] C. Lee, R. Rajkumar, and C. Mercer. Experiences with Processor Reservation and Dynamic QoS in Real-Time Mach. In *IEEE Multimedia Computing and Systems*, 1996.

[24] K.-W. Lee et al. An integrated source coding and congestion control framework for video streaming in the internet. In *INFOCOM (2)*, pages 747–756, 2000.

[25] B. Li and K. Nahrstedt. A control-based middleware framework for quality of service adaptation. *IEEE Journal on Selected Areas in Communications*, September 1999.

[26] S. Manne, A. Klauser, and D. Grunwald. Pipeline gating: Speculation control for energy reduction. In *25th Intl. Symposium on Computer Architecture*, 1998.

[27] S. Narayanswamy et al. A low-power lightweight unit to provide ubiquitous information access: Application and network support for infopad. *IEEE Personal Communications*, 1996.

[28] B. Noble. System support for mobile, adaptive applications. *IEEE Personal Communications*, 2000.

[29] C. Papadopoulos and G. Parulkar. Retransmission-based error control for continuous media applications. In *The 6th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, 1996.

[30] P. Pillai and K. G. Shin. Real-Time Dynamic Voltage Scaling for Low-Power Embedded Operating Systems. In *Proc. of the 18th ACM Symp. on Operating Systems Principles*, 2001.

[31] L. Qian et al. A General Joint Source-Channel Matching Method for Wireless Video Transmission. In *Proceedings of the Data Compression Conference*, 1999.

[32] R. Rajkumar et al. Resource kernels: a resource-centric approach to real-time and multimedia systems. In *SPIE Multimedia Computing and Networking*, 1998.

[33] I. Richardson and Y. Zhao. Adaptive algorithms for variable complexity video coding. In *Proceedings of ICIP*, 2000.

[34] D. G. Sachs et al. Hybrid arq for robust video streaming over wireless lans. In *Proceedings of the International Conference on Information Technology: Coding and Computing*, 2001.

[35] R. Sasanka, C. J. Hughes, and S. V. Adve. Combining Intra-Frame with Inter-Frame Hardware Adaptations to Save Energy. In *Proceedings of the Tenth International Conference on Architectural Support for Programming Languages and Operating Systems*, Oct. 2002. To appear.

[36] P. Sudame and B. Badrinath. On providing support for protocol adaptation in mobile wireless networks. *Mobile Networks and Applicatons*, 2001.

[37] M. Weiser et al. Scheduling for Reduced CPU Energy. In *Proc. of the 1st Symposium on Operating Systems Design and Implementation*, 1994.

[38] R. West and K. Schwan. Quality events: A flexible mechanism for quality of service management. In *7th IEEE Real-Time Technology and Applications Symposium*, 2001.

[39] W. Yuan and K. Nahrstedt. A middleware framework coordinating processor/power resource management for multimedia applications. In *Multimedia and Information Technology Symposium (MITS)*, 2001.

[40] W. Yuan and K. Nahrstedt. Integration of dynamic voltage scaling and soft real-time scheduling for open mobile systems. In *Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, 2002.

[41] M. Zorzi and R. Rao. Error control and energy consumption in communications for nomadic computing. *IEEE Transactions on Computers (Special Issue on Mobile Computing)*, 1997.