Energy-Aware System Design for Wireless Multimedia

Hans Van Antwerpen [*] Nikil Dutt [†] Rajesh Gupta [‡] Shivajit Mohapatra [§]
Cristiano Pereira [¶] Nalini Venkatasubramanian [‖] Ralph von Vignau [**]

ABSTRACT

In this paper, we present various challenges that arise in the delivery and exchange of multimedia information to mobile devices. Specifically, we focus on techniques for maintaining QoS to end-user multimedia applications (e.g. video streaming, multimedia conferencing) while maximizing device lifetimes. In order to cope with the resource intensive nature of multimedia applications (in terms of computation, bandwidth and consequently power) and dynamic congestion levels in wireless networks, an end-to-end approach to QoS-aware power optimization is required. We discuss the trend towards such an integrated approach that couples the architectural, OS, middleware and application layers to achieve both user experience and device energy gains. We conclude with a discussion of tools for integrated system design and testing that will aid in rapid deployment of wireless multimedia.

1 Motivation

Limiting the energy consumption of low-power mobile devices has become an important research objective in recent years. The capabilities of these devices are limited by their modest sizes and the finite lifetimes of the batteries that power them. As a result, minimizing the energy usage of every component (e.g. CPU, network card, display, architecture etc.) in such devices remains an important design goal and continues to pose significant challenges. On the other hand, rapid advances in processor and wireless networking technology are ushering in a new class of multimedia applications (e.g. video streaming/conferencing) for mobile handheld devices. Multimedia applications have distinctive Quality of Service(QoS) and processing requirements which tend to make them extremely resource-hungry. Moreover, the device specific attributes(e.g form factorof handhelds) significantly influence the human perception of multimedia quality. As a result delivering high quality realtime multimedia content to mobile handheld devices remains a difficult challenge.

These issues have been aggressively pursued by researchers and numerous interesting power optimization solutions have been proposed at various cross computational levels - system cache and external memory access optimizations [2, 11], dynamic voltage scaling(DVS) [18, 6] of the CPU, dynamic power management of disks and network interfaces(NICs) [7, 4], efficient compilers and application/middleware [14] based adaptations for power management. Interestingly, power optimization techniques developed for individual components of a device have remained seemingly incognizant of the strategies employed for other components. Therefore, increased research effort needs to be devoted to study the important issues involved in the interplay between the power management [21, 13] of the various components. While focussing their attention to a single component, researchers make a general assumption that no other power optimization schemes are operational for other components. Noticeably, the joint performance of an aggregation of techniques at various levels has received relatively little interest. The cumulative power gains observed by aggregating techniques at each stage can be potentially significant; however, it also requires a study of the trade-offs involved and the customizations required for unified operation.

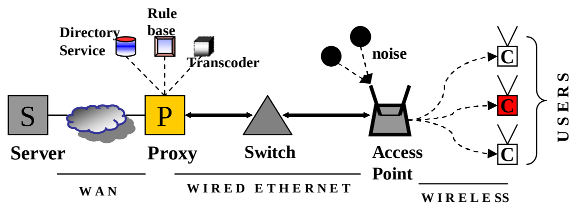

Fig. 1: **System Model**

System Model: We assume the system model depicted in Fig. 1. The system entities include a multimedia server, a proxy server that utilizes a directory service, a rule base for specific devices and a video transcoder, an ethernet switch, the wireless access point and users with low-power wireless devices. The multimedia servers store the multimedia content and stream videos to clients upon receipt of a request. The users issue requests for video streams on their handheld devices. All communication between the handheld device and the servers are routed through the proxy server, that can transcode the video stream in realtime. The middleware executes on both the handheld device and the proxy, and performs two important functions. On the device, it obtains residual energy availability information from the underlying architecture and feeds it back to the proxy and relates the video stream parameters and network related control information to lower abstraction layers. On the proxy, it performs a feedback based power aware admission control and realtime transcoding of the video stream, based on the feedback from the device. It also regulates the video transmission over the network based on the noise level and the video stream quality. Additionally, the middleware exploits dynamic global state information(e.g mobility info, noise level etc.) available at the directory service and static device specific knowledge (architecture, OS, video quality levels) from the static rule base, to optimally perform its functions. The rate at which feedbacks are sent by the device is dictated by administrative policies like periodic feedback etc.. Moreover

[*]Phillips Semiconductors
[†]dutt@ics.uci.edu - University of California, Irvine
[‡]rgupta@ucsd.edu - University of California, San Diego
[§]mopy@ics.uci.edu - University of California, Irvine
[¶]cpereira@cs.ucsd.edu - University of California, San Diego
[‖]nalini@ics.uci.edu - University of California, Irvine
[**]Phillips Semiconductors

we assume that network connectivity is maintained at all times.

The interaction between different layers is even more important in distributed applications where a combination of local and global information helps and improves the control decisions (power, performance and QoS trade-offs) made at runtime. Fig. 2 presents the different computation levels in a typical handheld computer and shows the cross layer interactions for optimal power and performance deliverance.

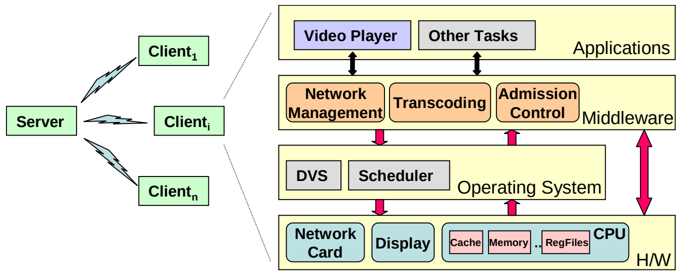

Fig. 2: **Abstraction Layers in Distributed Multimedia Streaming**

The purpose of our study is to develop and integrate hardware based architectural optimization techniques with high level operating system and middleware approaches (Fig. 2), for improvements in power savings and the overall user experience, in the context of video streaming to a low-power handheld device. Multimedia applications heavily utilize the biggest power consumers in modern computers: the *CPU*, the *network* and the *display*(see Fig. 2). Therefore, we aggregate hardware and software techniques that induce power savings for these resources. To maximize power gains for a CPU architecture, we identify the predominant internal units of the architecture that contribute to power consumption. We use higher-level knowledge such as quality and encoding parameters of the video stream to optimize internal cache configurations, CPU registers and the external memory accesses. Further we study the trade-offs of using DVS alongside the other optimizations. Knowledge of the underlying architectural configuration is used by the compiler to generate code that compliments the optimizations at the low-level architecture. Similarly, we utilize hardware/design level data(e.g cache config.) and user-level information(video quality perception) to optimize middleware and OS components for improved performance and power savings - through effective video transcoding, power-aware admission control and efficient network transmission. We reduce the power consumption of the network card by switching it to the "sleep" mode during periods of inactivity. An efficient middleware is used to control network traffic for optimal power management of the network interface. To maximize user experience, we conduct extensive tests to study video quality and power trade-offs for handheld computers. We use these results to drive our optimization efforts at each computing level.

In the rest of the paper, we present some of the research challenges encountered at each cross computational level and finally propose an integrated approach involving both distributed proxy based adaptations coupled with coordinated cross-layer energy optimizations at the device.

2 Hardware/Architectural Level Optimizations

Multiprocessor Systems-on-a-Chip (MPSoC) architectural platforms are increasingly being employed to solve a diverse spectrum of problems in the embedded and mobile systems domain. MPSoC architectural platforms typically employ multiple processor cores, together with specialized computational engines to meet the computational demands of the applications. Since most multimedia applications spend a significant amount of time accessing and transforming audio and video data, the design of the memory subsystem architecture, and compiler support for exploiting the specialized memory structures are critical for meeting the performance, power and cost budgets of such applications.

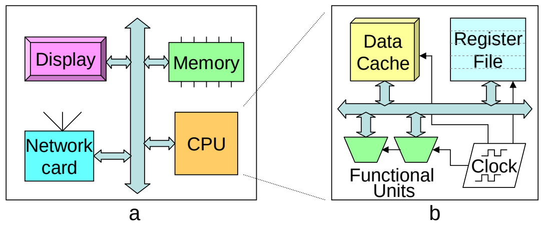

Fig. 3: **Main Components of a Handheld Device (a) and CPU Detail (b)**

Since the memory subsystem will dominate the cost (area), performance and power, we have to pay special attention to how the memory subsystem can benefit from customization. The memory can be selectively cached; the cache line size can be determined by the application; the designer can opt to discard the cache completely and choose specialized memory configurations such as FIFOs and stream buffers; and so on. The exploration space of different possible memory architectures is vast, and there have been attempts to automate or semi-automate this exploration process [9].

2.1 Hardware-level Knobs for Handheld Devices

As shown on Fig. 3(a), there are three major sources of power consumption in a handheld device (e.g. iPaq): display (around 1W for full backlight), network hardware (1.4W) and CPU/memory (1-3W, with the additional board circuits). Each of these subsystems expose ways for controlling the power dissipation. In case of the display (LCD), the main energy drain comes from the backlight, which is a predefined user setting and therefore has a limited degree of controllability by the system (without affecting the final utility). The network interface allows for efficient power savings if cognizant of the higher level protocol's behavior and will be explored in a subsequent section. Out of the three components mentioned above, the CPU coupled with the memory subsystem poses the biggest challenge. Its intrinsic high dependence on the input data to be processed, the quality of the code generated by the compiler and the organization of its internal architecture make predicting its power consumption profile very hard in general; nevertheless, very good power saving results can be obtained by utilizing the knowledge of the application running on it and through extensive profiling of a representative data input set from the application's domain. Over the rest of this section, we focus our attention on the possible optimizations at the CPU level for a multimedia streaming application (MPEG).

We identified the subcomponents of the CPU (Fig. 3(b)) that consume the most power and observed the power distribution inside the CPU for MPEG decoding. By running the decoder process in a power simulator (Wattch) for videos of various types and by measuring the relative power consumption of each unit in the CPU we generate the internal processor power distribution. We conclude that:

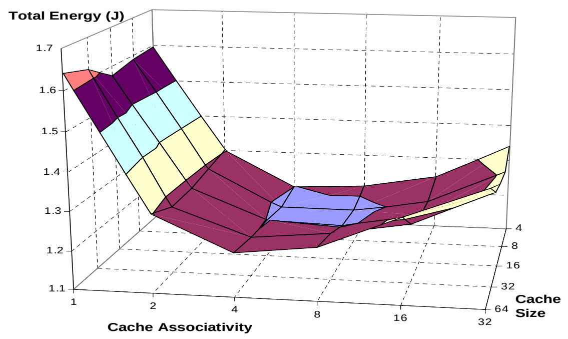

Fig. 4: **Cache Energy Variation on Size and Associativity**

• *The relative power contribution of the internal units of the CPU do not vary significantly with the nature or quality of the video played.* A possible reason for this is the symmetrical and repetitive nature of MPEG decoding, whose processing is done on fixed size blocks or macroblocks.

• *The units that show an important contribution to the overall power consumption and are amenable for power optimization are: caches, register files, functional units.* Cache behavior greatly affects the memory performance and hence power consumption, so we optimize the entire memory subsystem in an integrated way.

We briefly discuss the components identified above and suggest some additional improvements as a part of future work.

•**Caches/Memory**: cache configurations are determined by their size, number of sets, and associativity. The size specifies how large a cache should be, while the associativity/number of sets control its internal structure. We identify that most power gains for MPEG are possible through cache reconfiguration, more specific the data cache; cache optimizations influence memory traffic, so they are studied in an integrated way.

•**Frame Traversal**: Decompressing MPEG video in its implied order does not leave space for exploiting the limited locality existent between dependent macroblocks. By just changing the frame traversal order algorithm based on the existing locality, faster decompression rates and significant power saving are achieved via reduced memory accesses [5]. Our proxy-based approach allows for a transparent on-the fly traversal reordering at transcoding time, giving an advantage over previous work where this was done at the device, incurring unacceptable frame decoding delays.

We should mention that while current processors (including the ARM core on iPaq) in general do not exhibit such aggressive architectural reconfigurations, except for special purposes, there are many research projects on this and eventually the techniques will be incorporated into more future processors. Another knob, independent of the above is power management through the use of dynamic voltage scaling of the processor(DVS). DVS provides significant savings for MPEG streaming as it allows tradeoffs for transforming the frame decoding slack time (CPU idle time) into important power savings. We discuss DVS in a subsequent section and investigate the implications of DVS on other knobs in the system. All these knobs when fine-tuned for a specific video quality, will provide the best operating point(for power and performance) for a specific video stream.

2.2 Quality-driven Cache Reconfiguration

There are various techniques pertinent to cache optimizations. Power consumption for the cache depends on the runtime access counts: while hits result in only a cache access, misses add the penalty of accessing the main memory (external). Fortunately, in most applications the inherent locality of data means that cache miss rate is relatively low and so are accesses to external memory. However, MPEG decoding exhibits a relatively poor data locality, which, when combined with the large data sets exercised by the algorithm, leads to an increase in the cache memory-traffic.

In order to find the best solution point, we performed an extensive simulation and profiling with data that is representative for the video domain. Internal CPU caches are characterized by their size($\boldsymbol{S}$), number of sets($\boldsymbol{NS}$), line size($\boldsymbol{LS}$) and associativity($\boldsymbol{A}$). Over the next paragraphs, by cache we refer to data cache (not instruction cache, which is not the scope of our optimizations).

Our cache reconfiguration goal is optimizing energy consumption for a particular video quality level $\boldsymbol{Q_k}$. In general, cache power consumption for a particular configuration and video quality is given by the function $\boldsymbol{E_{cache,k}(S, A)}$. By profiling this function for the entire search space $\boldsymbol{(S, A)}$ of available cache configurations, we generate a cache energy variation graph shown in Fig. 4. Depending on the video quality $\boldsymbol{Q_k}$ played, there will be one optimal operating point for that video quality: $\boldsymbol{(S_k^{opt}, A_k^{opt})}$. We found out that for all video qualities an optimized operating point exists and it improves cache power consumption by up to 10-20% (as opposed to a suboptimized configuration). This technique effectively fine-tune the organization of the cache so that it perfectly matches the application and the data sets to be processed, yielding important power savings.

2.3 Reducing Backlight Power Consumption

As mentioned earlier, the backlight accounts for significant energy overheads in a low-power device. However, significant energy savings are possible by operating the device at a lower backlight intensity levels. We therefore explore a more aggressive approach to brightness compensation and device backlight control for streaming video. Furthermore, the adaptation is shifted away from the low-power device and performed at a network proxy server, obviating the need for the decoder on the device to be modified. We have found that aggressive brightness compensation is possible for streaming video as compared to still images, without considerably impacting the video quality. This is because small defects (introduced due to aggressive compensation) that might be noticeable in a still image are less discernable in streaming video where several frames (images) are displayed on the screen every second. We also propose an effective brightness compensation algorithm for optimized power savings [15]. Finally, we introduce middleware based adaptation schemes which integrate our compensation algorithm to achieve low power backlight operation for streaming video content to mobile handheld devices. Our proposed approach gives significant power reductions, up to 60% of the power consumption attributed to the backlight, depending on the chosen adaptation scheme and the characteristics of the streamed video.

We assume that the proxy server that has access to a database of profiled luminosity values for various video streams and device specific parameters (e.g. number of backlight levels, average luminosity at each level etc.), a rule base to determine compensation values and a video transcoder(Fig. 5); and low-power wireless devices capable of displaying streaming MPEG video content. Moreover, all communication between the handhelds and the multimedia server are routed through the proxy server that can change the video stream in real-time. Each device/client has an

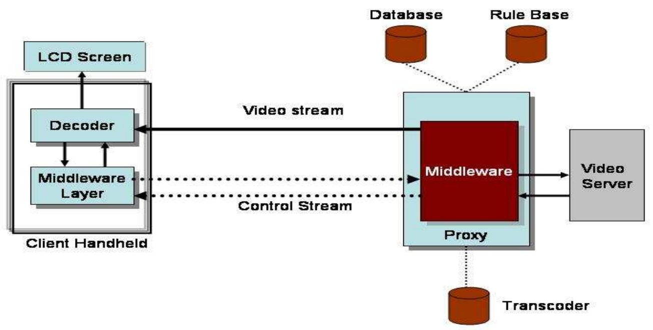

Fig. 5: **Model for Backlight Adaptation**

application layer where the video stream is decoded and a middleware layer which routes the information flowing from and to the video decoder application. The client middleware layer has access to system parameters such as the backlight levels, the current battery level and information identifying the type and make of the handheld (e.g. iPAQ, Jornada etc). In addition to accessing these system parameters, the middleware layer on the client can change these parameters (e.g. operating backlight level) through API calls to the underlying OS. The middleware on the proxy performs the dynamic adaptation of the streaming video content (brightness compensation) and communicates control information to the client middleware (operating backlight levels) through the low bandwidth control stream. The proxy maintains a database of information about the videos available at the server and information specific to different handheld types such as the number, luminous intensity and average power consumption of the backlight levels. Additionally, the proxy also employs a static rule base which specifies conditions which determine values for backlight and video compensation. The database and certain parameters of the rule base are populated by extensive profiling and subjective assessment of videos on different handhelds.

3 OS/Middleware Level Optimizations

As seen in the previous section, architectural level optimizations can lead to substantial power and performance improvements. The gains can be further amplified if the low-level architecture is cognizant of the exact characteristics of the streamed video. An adaptive middleware framework at a proxy can dynamically intercept and doctor a video stream to exactly match the video characteristics for which the target architecture has been optimized. Additionally, it can regulate the network traffic to induce maximal power savings in a network interface. Additionally, with knowledge of the video stream the operating system can employ an optimized dynamic voltage scaling of the CPU.

3.1 Integrated Dynamic Voltage Scaling

The previous section shows that significant power savings can be achieved by optimally reconfiguring the cache to match the video quality. The savings can be further increased when this is combined with dynamic voltage scaling (DVS) [6]. A processor normally operates at a specific supply voltage $\boldsymbol{V}$ and clock frequency $\boldsymbol{f}$. The dynamic power dissipated by the CPU (and any other CMOS circuits due to switching activity, in addition to the static component) varies linearly with frequency and circuit capacitance, and quadratically with voltage: $\boldsymbol{P \propto C \times f \times V^2}$. The disadvantage of applying dynamic voltage scaling is its power-performance tradeoff.

In MPEG decoding, frames are processed in a fraction of the frame delay ($\boldsymbol{F_d = 1/frame\_rate}$). The actual frame decoding time $\boldsymbol{D}$ depends on the type of MPEG frame being processed (**I**, **P**, **B**) and is also influenced by the cache configuration $\boldsymbol{(S, A)}$ and DVS setting $\boldsymbol{(f, V)}$. We assume a buffered based decoding, where the decoded frames are placed in a temporary buffer and are only read when the frame is displayed. This allows us to decouple the decoder from the displaying; decoding time it still different for different frame, but we can assume an average $\boldsymbol{D}$ for a particular video stream/quality. The difference between the average frame delay and actual frame decoding time gives us the slack time $\boldsymbol{\theta = F_d - D}$. When we perform DVS, we slow down the CPU so as to decrease the slack time to a minimum. Cache configuration also slightly influences the frame decoding time (due to the cache misses, which translate into external memory traffic), extreme values proving very inefficient. An optimized cache combined with DVS should yield best power saving results. Through simulation, we find the best operating points for the DVS/cache reconfiguration combined approach in a manner similar to the one applied in the previous section.

3.2 Power Aware Operating System Architecture

We view the notion of power awareness in the application and OS as a capability that enables a continuous dialogue between the application, the OS, and the underlying hardware. This dialogue establishes the functionality and performance expectations (or even contracts, as in real-time sense) within the available energy constraints. We describe here our implementation of a specific service, namely the task scheduler, that makes the OS power aware. The architecture is composed of two software layers and the OS kernel. One layer interfaces applications with operating system and the other layer makes power related hardware "knobs" available to the operating system. Both layers are connected by means of corresponding power aware operating system services as shown in Figure 6. At the topmost level, embedded applications call the API level interface functions to make use of a range of services that ultimately makes the application energy efficient in the context of its specific functionality. The API level is separated into two sub-layers. PA-API layer provides all the functions available to the applications, while the other layer provides access to operating system services and power aware modified operating system services (PA OS Services). Active entities that are not implemented within the OS kernel should also be implemented at this level (threads created with the sole purpose of assisting the power management of an operating system service.

We call this layer the power aware operating system layer (PA-OSL). To interface the modified operating system level and the underlying hardware level, we define a power aware hardware abstraction layer (PA-HAL). The PA-HAL gives the access to the power related hardware "knobs" in a way that makes it independent of the hardware.

3.3 Middleware based Network Traffic Regulation

In this section, we develop a proxy-based traffic regulation mechanism to reduce energy consumption by the device network interface. Our mechanism (a) dynamically adapts

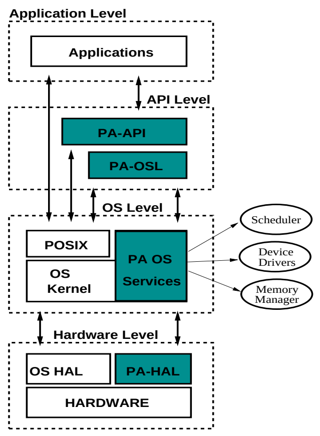

Fig. 6: Power Aware Operating System Architecture

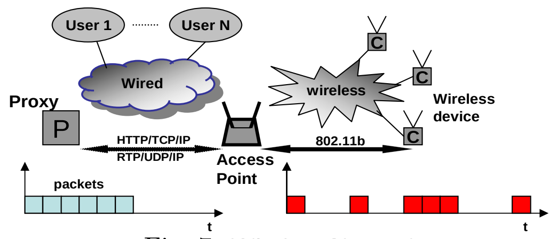

Fig. 7: **Wireless Network**

Quality	Parameters	Avg. Power (WinCE)	Avg. Power (Linux)
(Q8)	SIF, 30fps, 650Kbps	4.42W	6.07W
(Q7)	SIF, 25fps,450Kbps	4.37W	5.99W
(Q6)	SIF, 25fps, 350Kbs	4.31W	5.86W
(Q5)	HSIF, 24fps, 350Kbps	4.24W	5.81W
(Q4)	HSIF, 24fps, 200Kbps	4.15W	5.73W
(Q3)	HSIF, 24fps, 150Kbps	4.06W	5.63W
(Q2)	QSIF, 20fps, 150Kbps	3.95W	5.5W
(Q1)	QSIF, 20fps, 100kbps	3.88W	5.38W

Table 1: **Energy-Aware Transformations for Compaq Ipaq 3650 with bright backlight, Cisco 350 Series Aironet WNIC card. (Q1) Terrible, (Q2) Bad, (Q3) Poor, (Q4) Fair, (Q5) Good, (Q6) Very Good, (Q7) Excellent, (Q8) Like Original**

to changing network(e.g noise) and device conditions. (b) accounts for attributes of the wireless access points (e.g. buffering capabilities) and the underlying network protocol (e.g. packet size). (c) uses the proxy to buffer and transmit optimized bursts of video along with control information to the device. However, even though packets are transmitted in bursts by the proxy, the device receives packets that are skewed over time Fig. 7; this cuts power savings, as the net *sleep* time of the interface is reduced. The skew is caused due to the ethernet access protocol(e.g CSMA/CD) and/or the fair queueing algorithms implemented at the AP. Our mechanism optimizes the stream, such that the optimal video bursts sizes are sent for a given noise level, thus maximizing energy savings without performance costs.

Wireless network interface(WNIC) cards typically operate in four modes: *transmit, receive, sleep and idle.* We estimated the power consumption of the Cisco Aironet 350 series WLAN card to have the following power consumption characteristics: *transmit*(1.68W), *receive*(1.435W), *idle* (1.34W) and *sleep*(0.184W) which agree with the measurements made by Havinga et al. in [10]. This observation [3] suggests that significant energy savings can be achieved by transitioning the network interface from *idle* to *sleep* mode during periods of inactivity. The use of bursty traffic was first suggested by Chandra [3, 4] and control information was used for adaptation in [16].

We analyze the above power saving approach using a realistic network framework(Fig. 7), in the presence of noise and AP limitations. The proxy middleware buffers the transcoded video and transmits I seconds of video in a single burst along with the time $\tau = I$ for the next transmission as control information. The device then uses this control information to switch the interface to the active/idle mode at time $\tau + \gamma \times D_{EtoE}$, where γ is an estimate between zero and one and D_{EtoE} is the end-to-end network delay with no noise [16].

We acknowledge that a QoS aware preferential service algorithm at the access point can impact power management significantly. The above analysis can be used by an adaptive middleware to calculate an optimal I(burst length) for any given video stream and noise level. Note that energy overhead for buffering the video packets is not affected by using our strategy because the number of read and write memory operations remain unchanged irrespective of the memory buffer size.

4 Application Layer Adaptation

Improving the service lifetimes of low-power mobile devices through effective power management strategies can facilitate optimization of user experience for streaming video on to handheld devices. To achieve this, a system should be able to dynamically adapt to global system changes, such that the entire duration of a requested video is streamed to the user at the highest possible quality, while meeting the power constraints of the user's low-power device. We achieve such an optimal balance between power and performance, by introducing a notion of "*Utility Factor* U_F" for a system, and optimizing the U_F for the system. This approach precludes the system from aggressively optimizing for power at the expense of performance and vice-versa; thereby providing an optimized operating point for the system at all times. U_F is a measure of "user satisfaction" and we specify it as follows: given the residual energy E_{res} on a handheld device, a threshold video quality level ($Q_A : Q_{MAX} \geq Q_A \geq Q_{MIN}$) acceptable to the user, and the time of the video playback T, the U_F of the system is non-negative, if the system can stream the highest possible quality of video to the user such that the time, quality and the power constraints are satisfied; otherwise U_F is negative. Let P_{VID} denote the average power consumption rate of the video playback at the handheld and Q_{PLAY} be the quality of video(Table 1)streamed to the user by the system. Using the above notation, we define U_F as follows:

$$U_F = \begin{cases} Q_{PLAY} - Q_{MIN} & \mathit{IFF}\ P_{VID} * T < E_{RES} \\ & Q_{PLAY} \geq Q_A \\ -1 & \mathit{Otherwise} \end{cases}$$

5 Related Work and Conclusions

To provide acceptable video performance at the hardware level, efforts have concentrated on analyzing the behavior of the decoder software and devising either architectural enhancements or software improvements for the decoding algorithm. Until recently it was believed that caches can bring no potential benefit in the context of MPEG (video) decoding. In fact, due to the poor locality of the data stream, many MPEG implementations viewed video data as "un-cacheable" and completely disabled the internal caches during playback. However, Soderquist and Leeser [17] show that video data has sufficient locality that can be exploited to reduce cache-memory traffic by 50 percent or more through simple architectural changes. Dynamic Voltage Scaling [12, 6] for MPEG streams have been widely researched. A different way of improving cache performance by reordering frame traversal was proposed in [5]. Register file reconfiguration was applied in [2]. At the application and middleware levels, the primary focus has been to optimize network interface power consumption [7, 3, 4]. A thorough analysis of power consumption of wireless network interfaces has been presented in [7]. Chandra et al. [3] have explored the wireless network energy consumption of streaming video formats like Windows Media etc.. In [4], they have explored the effectiveness of energy aware traffic shaping closer to a mobile client. In [16], Shenoy suggests performing power friendly proxy based video transformations to reduce video quality in real-time for energy savings. They also suggest an intelligent network streaming strategy for saving power on the network interface. We have a similar approach, but we model a noisy channel. Caching streams of multiple qualities for efficient performance has been suggested in [8]. The GRACE project [21] professes the use of cross-layer adaptations for maximizing system utility. They suggest both coarse grained and fine grained tuning of parameters for optimal gains. In [20], a resource aware admission control and adaptation is suggested for multimedia applications for optimal CPU gains. Dynamic transcoding techniques have been studied in [1] and objective video quality assessment has been studied in [19].

Conclusions: We conclude that significant energy gains are achievable for low-power devices if a cross-layer communication framework is developed that allows for the various computation levels (architecture, OS, middleware, application) to interact dynamically. Moreover, proxy-based middleware adaptations that are cognizant of the architecture/OS level adaptations can significantly improve user experience for multimedia applications. User perception of video played a vital role in deciding the proxy-based video transformations and in identifying architectural tuning knobs. In practice, the widespread deployment of such a unified power management framework for mobile devices would require a set of APIs (programming interfaces) to be implemented at the various computational layers; this API should facilitate effective communication between the various levels. Recent approaches towards power management suggest a more open and flexible architecture for mobile devices that allows higher layers to make informed adaptations at lower layers and vice-versa. A prototype implementation of the framework is currently underway as a part of the *FORGE*(*http://www.ics.uci.edu/˜forge*) and *DYNAMO*(*http://dynamo.ics.uci.edu*) projects at University of California, Irvine.

Rapid deployment of the technology presented in this paper will require the cost effective design of development platform that facilitates integrated system design. Tools such as *Nx-Builder* from Philips will allow us to define design templates for integrated multimedia applications. A major focus in Nx-Builder will be on the reuse of verification suites at all stages of the development flow. The Nx-Builder verification capabilities will encompass Trans-actors for data insertion, automated regression testing and the automated compilation of individual IP test suites into verification frameworks. Additionally the use of standards such as System C, RTL, XML and SPIRIT will also enable reuse by multiple EDA vendors.

References

[1] Acharia, S., and B.C.Smith. Compressed Domain Transcoding of MPEG. In *ICMCS* (1998).

[2] Azevedo, A., Cornea, R., Issenin, I., Gupta, R., Dutt, N., Nicolau, A., and Veidenbaum, A. Architectural and compiler strategies for dynamic power management in the copper project. In *IWIA* (2001).

[3] Chandra, S. Wireless Network Interface Energy Consumption Implications of Popular Streaming Formats. In *MMCN-02.*

[4] Chandra, S., and Vahdat, A. Application-specific Network Management for Energy-aware Streaming of Popular Multimedia Formats. In *Usenix Annual Technical Conference* (June 2002).

[5] chi Feng, W., and Sechrest, S. Improving data caching for software mpeg video decompression. In *IS&T/SPIE Digital Video Compresssion: Algorithms and Technologies* (1996).

[6] Choi, K., Dantu, K., Chen, W.-C., and Pedram, M. Frame-Based Dynamic Voltage and Frequency Scaling for a MPEG Decoder. In *ICCAD 2000* (2002).

[7] Feeney, L., and Nilsson, M. Investigating the Energy Consumption of a Wireless Network Interface in an ad hoc Networking Environment. In *IEEE Infocom* (April 2001).

[8] Flinn, J., and Satyanarayanan, M. Energy-Aware Adaptations for Mobile Applications. In *In Symposium on Operating Systems Principles* (December 1999).

[9] Grun, P., Dutt, N., and Nicolau, A. "Memory architecture exploration for programmable embedded systems". In *Kluwer Academic Publishers, Norwell, MA 2003.*

[10] Havinga, P. J. M. *Mobile Multimedia Systems.* PhD thesis, University of Twente, Feb 2000.

[11] Hughes, C. J., Srinivasan, J., and Adve, S. V. Saving energy with architectural and frequency adaptations for multimedia applications. In *MICRO* (2001).

[12] Mesarina, M., and Turner, Y. A Reduced Energy Decoding of MPEG Streams. In *MMCN* (January 2002).

[13] Mohapatra, S., and Venkatasubramanian, N. PARM: Power-Aware Reconfigurable Middleware. In *ICDCS-03.*

[14] Noble, B. D., Satyanarayanan, M., D.Narayanan, J.E.Tilton, and Flinn, J. Agile Application-Aware Adaptation for Mobility. In *In Symposium on Operating Systems Principles* (October 1997).

[15] Pasricha, S., Mohapatra, S., and et. al. "Reducing backlight power consumption for streaming video applications on mobile handheld devices". In *ACM/IEEE/IFIP Workshop on Embedded Systems for Real-Time Multimedia, 2003.*

[16] Shenoy, P., and Radkov, P. Proxy-Assisted Power-Friendly Streaming to Mobile Devices. In *MMCN* (2003).

[17] Soderquist, P., and Leeser, M. Optimizing the data cache performance of a software MPEG-2 video decoder. In *ACM Multimedia* (1997), pp. 291–301.

[18] Weiser, M., Welch, B., Demers, A., and Shenker, S. Scheduling for Reduced CPU Energy. In *In Symposium on Operating Systems Design and Implementation* (1994).

[19] Winkler, S. Issues in vision modeling for perceptual video quality assessment. In *Signal Processing 78(2), 1999.* (1999).

[20] Yuan, W., and Nahrstedt, K. A Middleware Framework Coordinating Processor/Power Resource Management for Multimedia Applications. In *IEEE Globecom* (Nov 2001).

[21] Yuan, W., Nahrstedt, K., Adve, S., Jones, D., and Kravets, R. Design and Evaluation of a Cross-Layer Adaptation Framework for Mobile Multimedia Systems. In *MMCN-03* (2003).