Chapter 14

ENERGY-AWARE ADAPTATIONS FOR END-TO-END VIDEO STREAMING TO MOBILE HANDHELD DEVICES

Shivajit Mohapatra[1], Nalini Venkatasubramanian[1], Nikil Dutt[1],
Cristiano Pereira[2], Rajesh Gupta[2]
[1]*University of California, Irvine;*[2]*University of California, San Diego*

Abstract Optimizing user experience for streaming video applications on handheld devices is a significant research challenge. In this chapter, we propose an integrated end-to-end power management approach that unifies low level architectural optimizations(CPU, memory, registers), OS power-saving mechanisms(dynamic voltage scaling) and adaptive middleware techniques(admission control, transcoding, network traffic regulation). Specifically, we identify interaction parameters between the different levels and optimize them to reduce power consumption. With knowledge of device configurations, dynamic device parameters and changing system conditions, the middleware layer selects an appropriate video quality and fine tunes the architecture for optimized delivery of video. Performance results indicate that architectural optimizations that are cognizant of user level parameters(e.g. transcoded video quality) can provide energy gains as high as 57.5% for the CPU and memory, when compared to the baseline case that does not employ any energy optimization. Middleware adaptations to changing network noise levels can save as much as 70% of energy consumed by the wireless network interface. Our approach to multiple-level and end-to-end management of power/performance has been implemented in a framework, called FORGE. We show how FORGE can substantially enhance the user experience in a mobile multimedia application.

Keywords: low-power optimization, cross-layer adaptation, power-aware middleware, FORGE project

14.1 MOTIVATION

Limiting the energy consumption is an important design goal for mobile devices. Designers have explored techniques for minimizing energy usage of most components, from CPU, network, display to peripherals of a mobile system platform. 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 factor of 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.

The difficulty here is due to the fact that energy efficient delivery of media content with good quality attributes requires tradeoffs across various layers of system implementation and functionality - from application to system software to networking. Since the optimal energy conditions can change dynamically, these optimizations should also allow for dynamic adaption of system functionality and its performance. In order to dynamically adapt to device mobility, systems need to have a high degree of “network awareness" (e.g. congestion rates, mobility patterns etc.) and need to be cognizant of a constantly changing global system state. Efforts are underway to exploit multimedia specific characteristics to enable a range of energy optimization techniques that adapt to, and optimize for, changes in application data (video stream), OS/Hardware (CPU, Memory, Reconfigurable logic), network (congestion, noise, node mobility), residual energy (battery) and even the user environment (ambient light, sound).

These issues have been aggressively pursued by researchers and numerous interesting power optimization solutions have been proposed at various cross computational levels. For instance, a sampling of optimizations across design domains are: system cache and external memory access optimizations [1, 15], dynamic voltage scaling(DVS) [29, 4], of the CPU, dynamic power management of disks and network interfaces(NICs) [8, 3], efficient compilers and application/middleware based adaptations for power management [22]. Interestingly, power optimization techniques developed for individual components of a device have remained seemingly incognizant of the strategies employed for other 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. However, the cumulative power gains for incorporating multiple techniques can be potentially significant. This requires careful evaluation of the trade-offs involved and the customizations required for unified operation [21]. 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.

For the mobile multimedia applications, Fig. 14.1 presents the different computation levels in a typical handheld computer and shows the cross layer interactions for optimized power and performance deliverance.

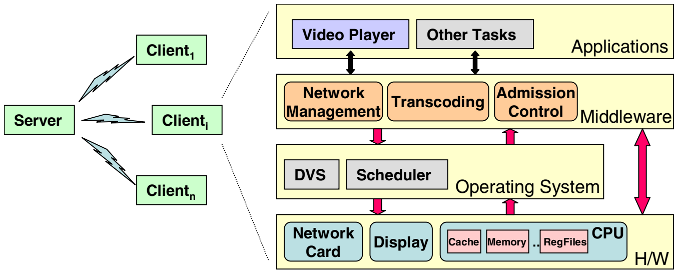

Figure 14.1. Abstraction Layers in Distributed Multimedia Streaming

The FORGE project aims to study the tradeoffs between power, performance and Quality of Service requirements across the various computational layers [6]. The goal of FORGE is to develop and integrate hardware based architectural optimization techniques with high level operating system and middleware approaches (Fig. 14.1), 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(Fig. 14.1). Therefore, in FORGE, we aggregate the hardware and software techniques that lead to 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 of the application such as quality and encoding parameters of the video stream to optimize internal cache configurations, CPU registers and the external memory accesses. Similarly, we utilize hardware/design level data (e.g. cache configuration) 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 the user experience, we have studied video quality and power trade-offs for handheld computers. These results drive our optimization efforts in FORGE at each computing level.

14.2 RELATED WORK

Let us briefly review the optimization techniques used at various levels, such as architecture, OS, middleware and application in the context of multimedia applications. We then examine the relationship of FORGE with prior and ongoing approaches in power aware middleware.

14.2.1 Architectural Adaptations

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 showed that video data has sufficient locality that can be exploited to reduce cache-memory traffic by 50 percent or more through simple architectural changes [28]. A different way of improving cache performance by reordering frame traversal was proposed in [9]. Register file reconfiguration was applied in [1]. [16] proposes a technique for combining two hardware adaptations (architecture adaptation and dynamic voltage scaling) to reduce energy in multimedia workloads. The algorithm presented chooses between one of the two adaptations or a combination, depending on their relative performance. This approach is similar to ours, in that architectural optimizations are combined with dynamic voltage scaling (DVS). However, instead of a frame-based adaption founded on profiling and prediction, we target tuning an architecture through available architectural parameters to specific video quality requirements. We apply the optimizations globally (for the entire period that a media of constant quality levels are played), rather than at frame granularity.

14.2.2 Operating System & Middleware Adaptations

Most power optimization efforts at the operating system level, have been focussed on techniques like dynamic voltage scaling(DVS) [29, 25, 18], and dynamic power management (DPM) [17, 7]. DVS exploits the fact the CMOS logic used in most current processors has a voltage dependent maximum operating frequency. So when used at lower frequencies, the processor can operate at a correspondingly lower voltage, thereby saving battery power. The challenge here is to accurately predict workload execution times for future jobs. While workloads can be predicted heuristically for best-effort applications [29], or based on worst case execution times of real-time applications [25], worst-case based approaches will almost certainly result in sub-optimal solutions, whereas heuristic predictions can cause timing violations for multimedia tasks. In the GRACE project, the authors suggest using an aggregate statistical demand of applications to adjust frequency/voltage for the processor [31]. DVS techniques for reducing energy in MPEG decoding has been studied in [20]. Additionally, scheduling techniques like DSRT [5] have been studied to deliver real-time guarantees.

At the OS/middleware levels, another primary focus has been to optimize network interface power consumption [8, 2, 3]. A thorough analysis of power consumption of wireless network interfaces has been presented in [8]. ECOSystem [32] is an OS level prototype that incorporates energy allocation and accounting mechanisms for various power consuming devices. ECOSystem uses the Currentcy [33] model which is an abstraction for formulating energy aware policies.

Chandra et al. [2] have explored the wireless network energy consumption of streaming video formats like Windows Media, Real media and Apple Quick Time. Chandra and Vahdat have explored the effectiveness of energy aware traffic shaping closer to a mobile client [3]. In [26], 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. FORGE uses a similar approach, but models a noisy channel. Caching streams of multiple qualities for efficient performance has been suggested in [10].

PowerScope [12] is an interesting tool that maps energy consumption to program structure. It first profiles the power consumption and system activity of a computer and then generates an energy profile from this data. Odyssey [22] presents an applications aware adaptation scheme for mobile applications. In this approach the system monitors resource levels, enforces resource allocation and provides feedback to the applications. The application then decides on the best possible adaptation strategy. In our approach we try to integrate the the positive aspects of all the three levels: OS, middleware and application. Application based adaptation will therefore enhance the performance of our framework. However, applications have to be specifically designed for the framework. JouleTrack [27] is a web-based energy measurement tool for profiling software energy consumption of applications based on StrongArm processor.

14.2.3 Cross-Layer Adaptation Frameworks

For efficient coordination and management of cross-layer adaptations, it is crucial to develop efficient resource allocation mechanisms. Q-RAM [19] models QoS management as a constraint optimization problem for maximizing system utility while guaranteeing minimum resources to each application. Puppeteer [11] presents a middleware framework that uses transcoding to achieve energy gains. Using the well defined interface of applications, the framework presents a distilled version of the application to the user, in order to draw energy gains. EQoS [24] formulates energy-aware QoS adaptation as a constraint optimization problem and solves it using heuristic algorithms.

The GRACE project [31, 30] uses cross-layer adaptations for maximizing system utility at lower energy costs. They suggest both coarse grained and fine grained tuning through global co-ordination and local adaptation of hardware, OS and application layers. The coarse/global adaptations are expensive and less frequent and occur only when global system changes are triggered (e.g task-set changes). The local adaptations are for the local variation in the execution of tasks. In GRACE, the global and local coordinators exist on the local device and perform the necessary adaptations. GRACE first tries to deliver highest utility for each application and then optimizes the energy using dynamic voltage scaling. In contrast, FORGE uses a proxy based distributed middleware approach, that integrates cross-layer(architecture, OS, middleware, application) adaptations on the local device with distributed adaptations such as adaptive traffic shaping and transcoding at the proxy for energy gains. While adaptations in GRACE are limited to the local mobile device, our framework design uses a distributed middleware layer to exploit global system knowledge (e.g. device mobility patterns, network noise levels etc.) to facilitate effective power management (e.g. wireless NIC). Moreover, we adopt an end-to-end approach to power optimization, where residual battery power of a mobile device also drives the adaptations. GRACE on the other hand provides a best-effort approach to energy optimization. Additionally, FORGE tries to tune architectural level parameters (e.g. cache configurations) to perform optimally for the currently executing application. The distributed middleware co-ordinates the adaptations at each level based on a rule-base and control information from the proxy.

14.3 SYSTEM MODEL

Our system model for a wireless mobile multimedia distributed system is shown in Fig. 14.2. 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 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 feedback are sent by the device is dictated by administrative policies (e.g. periodic feedback). Moreover, we assume that network connectivity is maintained at all times.

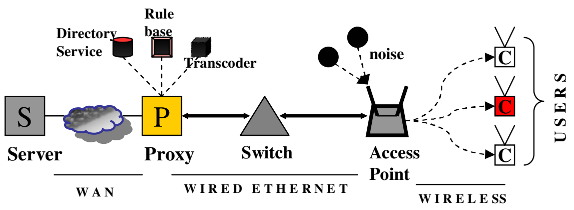

Figure 14.2. System Model

In the rest of the chapter, we present important research challenges encountered at each level and discuss approaches that involve both distributed proxy based adaptations coupled with coordinated cross-layer energy optimizations at the device.

14.4 HARDWARE/ARCHITECTURAL LEVEL OPTIMIZATIONS

The architectural optimizations are particularly important because of the use of microelectronic system-on-chip components used in multimedia platforms. 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.

Since the memory subsystem will dominate the cost (area), performance and power, we have to pay special attention to how it can benefit from customization. For example, 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. The exploration space of different possible memory architectures is vast, and there have been attempts to automate or semi-automate this exploration process [13].

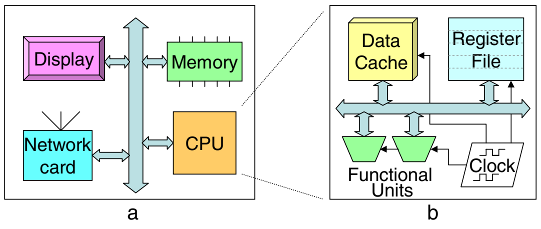

Figure 14.3. Main Components of a Handheld Device (a) and CPU Detail (b)

14.4.1 Hardware-level Optimizations for Handheld Devices

There are three major sources of power consumption in a handheld device such as a Compaq iPAQ 3650 for which we indicate the corresponding power numbers: display (approximately 1W for full backlight), network hardware (1.4W) and CPU/memory (1-3W, with the additional board circuits). Each of these subsystems also provide opportunities 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. The 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. In the rest of this section, we focus our attention on the possible optimizations at the CPU level for a multimedia streaming application (e.g MPEG-1).

We identified the subcomponents of the CPU (Fig. 14.3(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:

• 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 and 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 these components, their impact on overall power consumption and how it can be affected by these architectural choices:

• **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 reconfiguration of the data cache and its effect on the memory traffic, thus amplifying the effect of power optimizations through cache reconfiguration.

• **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 higher power savings are achieved via reduced memory accesses [9]. Our proxy-based approach allows for a transparent on-the fly traversal reordering at the proxy server. In addition dynamic voltage scaling provides further 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 and investigate the implications of DVS on other power optimizations in the system. All these parameters when fine-tuned for a specific video quality, will provide the best operating point(for power and performance) for a specific video stream.

14.4.2 Quality-driven Cache Reconfiguration

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 resort to extensive simulation and profiling with data that is representative of the video domain. Internal CPU caches are characterized by their size($\boldsymbol{S}$), number of sets($\boldsymbol{NS}$), line size($\boldsymbol{LS}$) and associativity($\boldsymbol{A}$). 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

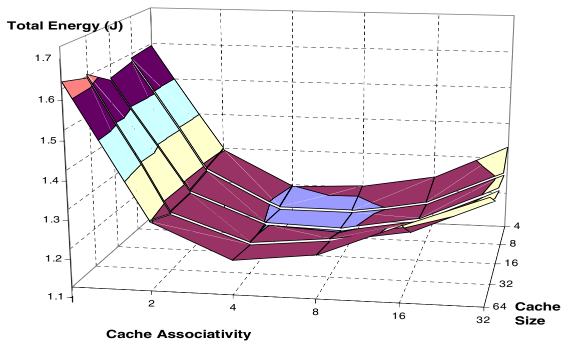

Figure 14.4. Cache Energy Variation on Size and Associativity

configurations, we generate a cache energy variation graph shown in Fig. 14.4. Depending on the video quality Q_k played, there will be one optimal operating point for that video quality: (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-tunes the organization of the cache so that it perfectly matches the application and the data sets to be processed, yielding important power savings.

14.5 OS/MIDDLEWARE LEVEL OPTIMIZATIONS

Gains in power reduction and performance improvement from architectural optimizations can be further amplified if the low-level architecture is cognizant of the exact characteristics of the streamed video. An adaptive middleware software 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. It can also 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.

14.5.1 Integrated Dynamic Voltage Scaling

For a given supply voltage, V, and clock frequency f, the dynamic power due to digital CMOS varies linearly with frequency and quadratically with the supply voltage (which is also the switching voltage). This relationship can be used at the application level [4]. In our case, for MPEG decoding, frames are processed in a fraction of the frame delay ($F_d = 1/frame\_rate$). The actual frame decoding time D depends on the type of MPEG frame being processed (**I**, **P**, **B**) and is also influenced by the cache configuration (S, A) and DVS setting (f, V). We assume a buffer 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 decoding of the frame from the displaying part; decoding time is still different for different frames, we therefore assume an average D for a particular video stream/quality. The difference between the average frame delay and actual frame decoding time gives us the slack time $\theta = F_d - D$. We can then perform DVS, where we slow down the CPU to use up the slack time. 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 yields the best power saving results. Determination of the best operating point for the DVS/cache reconfiguration requires simulation of the application with the power aware system software that has direct influence on the technology parameters. This is discussed next.

14.5.2 Power Aware Operating System Architecture

We view the notion of power 'awareness' in the application and OS as a capability to carry out 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 scheduler 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 14.5. 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. The 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 are 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 provides access to the power related hardware parameters in a way that makes it independent of the hardware.

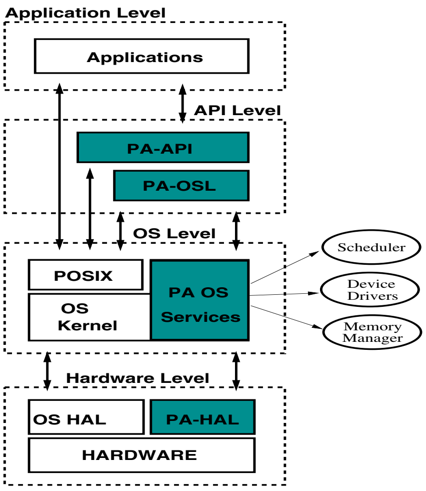

Figure 14.5. Power Aware Operating System Architecture

14.5.3 Middleware based Network Traffic Regulation

We now describe a proxy-based traffic regulation mechanism to reduce energy consumption by the device network interface. Our mechanism (a) dynamically adapts to changing network(e.g noise) and device conditions(e.g. residual battery energy). (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. 14.6; 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 wireless access points. Our mechanism optimizes the stream, such that 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 [14]. This observation suggests that considerable 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 [2, 3] and control information was used for adaptation in [26].

We analyze the above power saving approach using a realistic network framework(Fig. 14.6), in the presence of noise and AP limitations [21]. The proxy middleware buffers the transcoded video and transmits I seconds of video in a single burst along with the time τ=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.

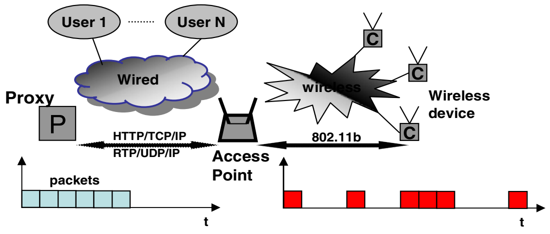

Figure 14.6. Wireless Network

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.

In the previous section, we demonstrated how low level architecture can be optimized using high level information. In this section, we presented two middleware techniques that can be used to compliment the low-level hardware optimizations, lower energy consumption of the NIC and improve the overall utility of the system. We now introduce a middleware based adaptation scheme for backlight power savings in handheld devices.

14.5.4 Reducing Backlight Power Consumption

The backlight accounts for considerable energy overheads in a low-power device. However, potentially large energy savings are realizable by operating the device at a lower backlight intensity levels. We 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 [23]. In this approach, 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 experiments indicate that this approach can provide power reductions of 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 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. 14.7); and low-power wireless devices capable of displaying streaming MPEG video content. All communication between the handhelds and the multimedia server are routed through the proxy server that can change the video stream in real-time.

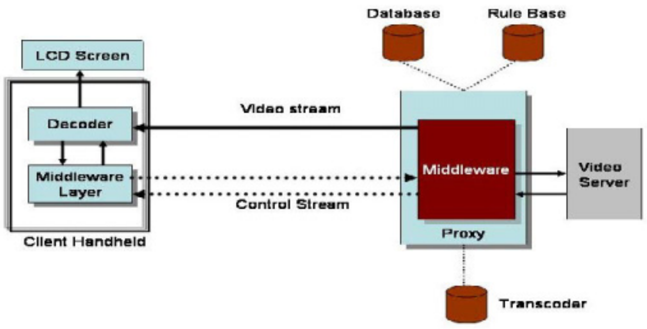

Figure 14.7. Model for Backlight Adaptation

Each device/client has an application layer where the video stream is decoded and a middleware layer which routes the information flowing to and from 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.

14.6 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, so 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 streamed to the user by the system. Using the above notation, we define U_F as follows:

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

Our experiments to determine video transcoding levels that affect the video quality against increased energy consumption indicate the following major conclusions:

● It is hard to programmatically identify video quality parameters(a combination of bit rate, frame rate and video resolution) that produced a user perceptible change in video quality and/or a noticeable shift in power consumption in handhelds.

● For all the video streams on handheld devices, it was enough to use just three standard intermediate formats(e.g SIF(320x240), Half SIF(340x160) and Quarter SIF(160x120)) for frame resolution values. Other resolutions did not

Table 14.1. 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

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

produce a perceptible quality change or power uptake compared to the nearest SIF encoded video with similar bit and frame rates.

Based on these conclusions, we identified eight dynamic video stream transformation parameters (Table 14.1) for our proxy-based realtime transcoding and use the profiled average power consumption values to perform our adaptations.

14.7 SUMMARY

It has been pointed out by several researchers that power optimization across various levels of system functionality and implementation (architecture, OS, middleware, application) can lead to much greater savings than the case when these are individually optimized for power. The challenge is how these optimizations can be coordinated across layers; what is the right architectural framework that allows this optimization to occur simultaneously and even dynamically? To answer this question, this paper proposes a proxy-based middleware solution to accommodate optimizations across diverse clients with limited computation and battery power by controlling the amount of needed computation and communication to the client device. We showed how such adaptation in the middleware can be used to improve energy efficient delivery of multimedia content in the case of streaming video. User perception of video also plays a vital role in deciding the proxy-based video transformations and in identifying architectural tuning “knobs". However, identifying the various video qualities remains a highly subjective aspect of the study. Identifying video quality levels objectively/programmatically still remains an open research challenge. In practice however, 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*) project.

Acknowledgments

This work was supported by funding from an ONR MURI Grant N00014-02-1-0715 and NSF NGS award ACI-0204028.

References

[1] Azevedo, Ana, Cornea, Radu, Issenin, Ilya, Gupta, Rajesh, Dutt, Nikil, Nicolau, Alex, and Veidenbaum, Alex (2001). Architectural and compiler strategies for dynamic power management in the copper project. In *International Workshop on Innovative Architecure*.

[2] Chandra, Surendar (2002). Wireless Network Interface Energy Consumption Implications of Popular Streaming Formats. In *Multimedia Computing and Networking*.

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

[4] Choi, Kihwan, Dantu, Karthik, Chen, Wei-Chung, and Pedram, Massoud (2002). Frame-Based Dynamic Voltage and Frequency Scaling for a MPEG Decoder. In *International Conference on Computer Aided Design*.

[5] Chu, H. H. and Nahrstedt, Klara (1999). "Cpu Service classes for multimedia applications". In *International Conference on Multimedia Computing and Systems*.

[6] Cornea, Radu, Dutt, Nikil, Gupta, Rajesh, Mohapatra, Shivajit, and et. al. (2003). ServiceFORGE: A Software Architecture for Power and Quality Aware Services. In *FME Symposium*.

[7] Douglis, Fred, Krishnan, P., and Marsh, B. (1994). Thwarting the power hungry disk. In *WINTER USENIX conference*.

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

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

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

[11] Flinn, Jason, de Lara, Eyal, Satyanarayanan, M., Wallach, Dan S., and Zwaenepoel, Willy (2001). "Reducing the energy usage of office applications". In *IFIP/ACM International Conference on Distributed Systems Platforms*.

[12] Flinn, Jason and Satyanarayanan, M. (1999b). PowerScope: a tool for profiling the energy usage of mobile applications. In *Second IEEE Workshop on Mobile Computing Systems and Applications*.

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

[14] Havinga, Paul J. M. (2000). *Mobile Multimedia Systems*. PhD thesis, University of Twente.

[15] Hughes, C. J., Srinivasan, J., and Adve, S. V. (2001a). Saving energy with architectural and frequency adaptations for multimedia applications. In *IEEE/ACM International Symposium on Microarchitecture*.

[16] Hughes, C. J., Srinivasan, J., and Adve, S. V. (2001b). Saving energy with architectural and frequency adaptations for multimedia applications. In *IEEE/ACM International Symposium on Microarchitecture*.

[17] Irani, S., Shukla, S., and Gupta, R. (2002). "Competitive analysis of dynamic power management strategies for systems with multiple power saving states". In *Design Automation and Test in Europe*.

[18] Kumar, Pavan and Srivastava, Mani (2000). Predictive Strategies for Low-Power RTOS Scheduling. In *International Conference on Computer Design*.

[19] Lee, C., Lehoczky, J., Siewiorek, D., Rajkumar, R., and et.al (1999). A Scalable solution to the multi-resource QoS problem. In *Real-Time Systems Symposium*.

[20] Mesarina, M. and Turner, Y. (2002). A Reduced Energy Decoding of MPEG Streams. In *Multimedia Computing and Networking*.

[21] Mohapatra, Shivajit, Cornea, Radu, Dutt, Nikil, Nicolau, Alex, and Venkatasubramanian, Nalini (2003). Integrated power management for video streaming to mobile handheld devices. In *ACMMM*.

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

[23] Pasricha, Sudeep, Mohapatra, Shivajit, and et. al. (2003). "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.*

[24] Pillai, P., Huang, H., and Shin, K. G. (2003). Energy-Aware quality of Service adaptation. In *Technical Report CSE-TR-479-03, Univ. of Michigan.*

[25] Pillai, P. and Shin, K. G. (2001). Real-Time Dynamic Voltage Scaling for Low-Power Embedded Operating Systems. In *In Symposium on Operating Systems Principles.*

[26] Shenoy, Prashant and Radkov, Peter (2003). Proxy-Assisted Power-Friendly Streaming to Mobile Devices. In *Multimedia Computing and Networking.*

[27] Sinha, Amit and Chandrakasan, Anantha (2001). "jouletrack - a web based tool for software energy profiling". In *Design Automation Conference.*

[28] Soderquist, Peter and Leeser, Miriam (1997). Optimizing the data cache performance of a software MPEG-2 video decoder. In *ACM Multimedia*, pages 291–301.

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

[30] Yuan, W. and Nahrstedt, K. (2004). Process Group Management in Cross-Layer Adaptation. In *Multimedia Computing and Networking.*

[31] Yuan, W., Nahrstedt, K., Adve, S., Jones, Doug., and Kravets, Robin (2003). Design and Evaluation of a Cross-Layer Adaptation Framework for Mobile Multimedia Systems. In *Multimedia Computing and Networking.*

[32] Zeng, H., Ellis, C., Lebeck, A., and Vahdat, A. (2002). "Ecosystem: Managing energy as a first class operating system resource". In *Architectural Support for Programming Languages and Operating Systems.*

[33] Zeng, H., Ellis, C., Lebeck, A., and Vahdat, A. (2003). Currentcy: Unifying policies for resource management. In *USENIX.*