A Cross-Layer Approach for Power-Performance Optimization in Distributed Mobile Systems*

Shivajit Mohapatra†, Radu Cornea†, Hyunok Oh†, Kyoungwoo Lee†, Minyoung Kim†,
Nikil Dutt†, Rajesh Gupta§, Alex Nicolau†, Sandeep Shukla¶, Nalini Venkatasubramanian†

† School of Information and Computer Science, UC Irvine,
§ Department of Computer Science and Engineering, UC San Diego,
¶ Department of Electrical and Computer Engineering, Virginia Tech.

Abstract

The next generation of mobile systems with multimedia processing capabilities and wireless connectivity will be increasingly deployed in highly dynamic and distributed environments for multimedia playback and delivery (e.g. video streaming, multimedia conferencing). The challenge is to meet the heavy resource demands of multimedia applications under the stringent energy, computational, and bandwidth constraints of mobile systems, while constantly adapting to the global state changes of the distributed environment. In this paper, we present our initiatives under the FORGE framework to address the issue of delivering high quality multimedia content in mobile environments. In order to cope with the resource intensive nature of multimedia applications and dynamically changing global state (e.g. node mobility, network congestion), an end-to-end approach to QoS aware power optimization is required. We present a framework for coordinating energy optimizing strategies across various layers of system implementation and functionality and discuss techniques that can be employed to achieve energy gains for mobile multimedia systems.

1. Introduction

Current trends indicate that delivery of multimedia content to mobile systems operating in distributed environments will drive many future applications. Distributed multimedia applications (e.g. video streaming) with their distinctive Quality of Service(QoS) and heavy resource demands can quickly drain the battery energy of mobile systems. Therefore, optimizing the energy consumption is an important design goal for such systems.

* This work was partially supported by NSF award ACI-0204028.

Researchers have proposed several techniques at various system levels (hardware, OS, network, application) for saving battery energy in mobile systems. However, power optimization techniques developed for individual components of a device (single system level) have remained seemingly incognizant of the strategies employed for other components. While focussing their attention to a single level, researchers make a general assumption that no other power optimization schemes are operational at other levels. We believe that the cumulative power gains for incorporating multiple techniques can be potentially significant, but this also requires careful evaluation of the trade-offs involved and the customizations required for unified operation [15].

Energy efficient delivery of multimedia content with good quality attributes, requires tradeoffs across various layers of system implementation and functionality - from application to system software to networking and distributed adaptation. 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.

Therefore, within the context of the FORGE project, we have concentrated our efforts on exploiting 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) and user preferences (preferred quality). The interaction between different system layers (Fig. 1) 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.

Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05)
1530-2075/05 $ 20.00 **IEEE**

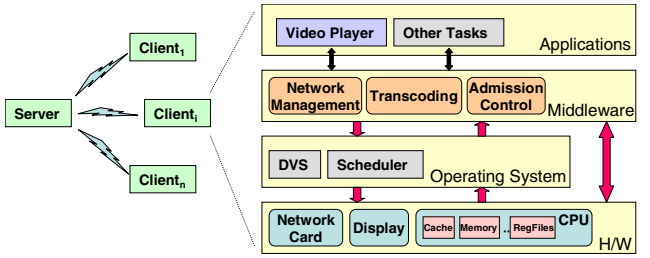

Figure 1. Abstraction Layers in Distributed Multimedia Streaming

The FORGE project [9] aims to study the tradeoffs between power, performance and QoS requirements across the various computational layers. The goal is to develop and coordinate application, middleware, OS and hardware energy optimization approaches, for improvements in power savings and the overall user experience, in the context of distributed multimedia applications. Multimedia applications heavily utilize the biggest power consumers in modern computers: the CPU, the network and the display. Therefore, in FORGE, we aggregate the hardware and software techniques that lead to power savings for these resources. As a result of the FORGE initiative, we have explored different ideas for achieving coordinated power management across different levels of system functionality in the context of multimedia applications.

- We investigate the use of an adaptive, distributed middleware framework (called "Dynamo") that coordinates global (proxy based) and local adaptations for power management in mobile devices operating in distributed environments.
- We have studied how to annotate application data with specific information that can be exploited during runtime at different levels of abstraction (hardware through application), for improving power efficiency and ultimately the user experience.
- Finally, we examine the tradeoffs of parameters defined in multiple layers such as application layer, OS layer network layer and hardware layer. For example, we consider image quality in application layer, compressed size in network layer, and execution time and power consumption in hardware layer. We investigate tradeoff between the parameters and system performance and optimize each parameter to minimize power consumption with meeting given image quality and network bandwidth constraints.

We assume the system model depicted in Fig. 2. The system entities include a multimedia server, a proxy node that can perform various optimizations on the stream (e.g. transcoding), the users with low-power wireless devices and

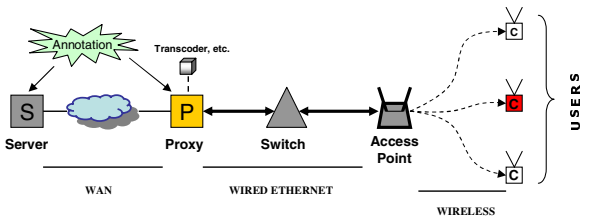

Figure 2. System Model

other network equipment along the way. The multimedia servers store media content and stream videos to clients upon requests issued by the users on their handheld devices. The communication between the handheld device and the servers can be routed through a proxy server – a high-end machine that has the ability to process the video stream in real-time. The proxy node can also perform inline profiling/annotation for real-time video streaming (videoconferencing is an example of such an application).

2. Cross-Layer Adaptation using the Dynamo Middleware Framework

In order to coordinate power management strategies across various system layers, one specific approach we have adopted is to use a distributed middleware framework to manage and interface the cross layer adaptations. A middleware framework specifically designed to address the cross-layer adaptation is effective because (i) the middleware can exploit global system state information (e.g. network noise, mobility patterns etc.) which are typically not available locally on a low-power device to drive power management strategies and (ii) it can abstract system level details effectively from higher layers (iii) coordinate local and remote adaptations by interfacing with the applications, network, OS and hardware.

We have designed and implemented an adaptive power-aware middleware framework called "Dynamo" that coordinates application adaptations, OS power-saving mechanisms (dynamic voltage scaling) and adaptive middleware techniques (energy-aware admission control, transcoding, network traffic regulation, mobility etc.) for improving performance and energy gains for mobile systems. We have identified interaction parameters between the different computational levels that can facilitate effective cross-layer coordinations. To deploy such an unified power management framework for mobile devices, we have developed a set of APIs at the various computational layers in order to establish a continuous dialogue between the various layers. Such a capability can be effectively exploited to drive various dynamic cross layer energy adaptations, either due to changes in global system state or due to changes in the local device state (e.g. battery energy).

As shown in Fig. 1, we assume that the mobile device has four levels - application, middleware, OS and hardware. The energy consuming components (e.g. cpu, NIC, LCD display) are at the hardware level. The next higher level is the operating system, which has access to the physical devices through well defined driver interfaces. We enhance the operating system for supporting cross-layer energy management. Next in the hierarchy is the distributed middleware level. Our framework entirely adds this layer for coordinating global and local adaptations. The middleware layer can be considered to have three abstract components to it - a "system" component that resides within the OS (with possible OS level code integration), a "network" component that implements a communication protocol to talk with a distributed proxy server and a "user level" component that performs the various middleware based adaptations using the information gathered from the network, the OS and the user (or application). The application layer provides user profiles (specific user preferences) and application specific context to the middleware and also performs dynamic QoS negotiations.

In this model, the middleware executes on both the mobile device and the proxy, and performs several important functions. On the device, the middleware makes the executing mobile applications both "*local state aware*" and "*global state aware*". By *local state awareness* we refer to everything that resides on the physical device the application is executing on - for example residual battery information, backlight settings, operating CPU frequency, memory information, other executing applications etc.. We also exploit our middleware to make individual system layers aware of the power management functionalities in the other layers. On the other hand, *global awareness* refers to information that is not available on the local device (e.g. bandwidth availability, network congestion, mobility information).

While most current power management strategies exhibit a certain degree of local awareness in their approach, none of the power management techniques have tried to exploit the information available in both global and local contexts. Based on the residual battery energy level (local context) of a device, a proxy can perform several energy aware adaptations. For example, the quality of a video stream can be dynamically reduced if the device running low on battery. The distributed middleware plays the role of both exposing global and local contexts to applications and proxies. Additionally, the middleware can use its cognizance of global/local states to provide valuable hints that aid the other system levels to better tune their adaptations.

Fig. 3 shows the various implemented components of our integrated cross-layer adaptation framework. The top level consists of the various applications that link to our framework. The primary components of the distributed middleware layer are the system/energy monitors, the adaptation manager and the communication manager. We implement a power-aware API interface for applications & middleware

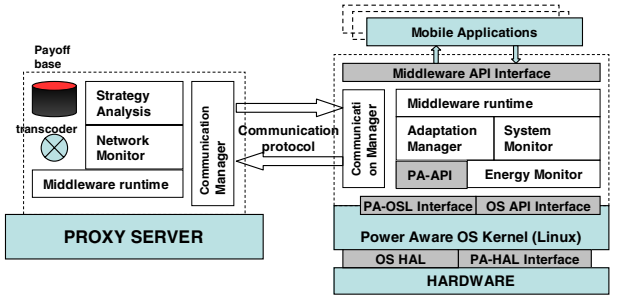

Figure 3. Framework Implementation

components to interface with the power aware operating system. The energy/system monitors are middleware libraries that export calls to provide applications (as well as other middleware modules) with local state information. The monitor interface has been implemented on Linux using the "proc" file system interface provided by linux [3]. The communication manager implements a middleware communication protocol to communicate state and control information between the device and proxy. The adaptation manager performs various middleware level adaptations (e.g strategy selection, QoS negotiation etc). The middleware employs an active runtime component (thread), that performs various background tasks (like updating local/global state periodically etc.). However, the functions of both the adaptation manager and the runtime are dictated by adaptations specific to applications.

In the power-aware operating system we implement the dynamic frequency scaling of the CPU. We have modified the Familiar Linux kernel v2.4 to implement our frequency scaling policies based on the rate-monotonic criterion [17]. Additionally, the kernel exports a set of API calls that enable the middleware/applications to set task execution parameters for dynamic adaptation. At the lowest level, we have a power aware hardware abstraction layer that implements code for accessing and changing power and performance knobs of the actual devices. We have developed wrapper functions that expose these capabilities to the middleware layer [1].

We studied the performance of the Dynamo framework using a streaming video application. We show that by using our framework, we were able to improve the overall video quality (user satisfaction) of the system over time within the energy budget of the mobile device. Fig. 4 plots the video quality (utility factor U_F) over time for a two hour video sequence with different initial battery lifetimes at the device. The horizontal lines indicate the without the cross-layer optimizations. In each case, we were able to manifest our energy savings due to cross-layer adaptations as improvements in the U_F, and hence the user experience. Fig. 5 shows the same video requested but this time with a random user induced energy cost (noise due to other processes, backlight changes etc.). Clearly, the utility factor is improved with the integrated approach even in the presence of noise.

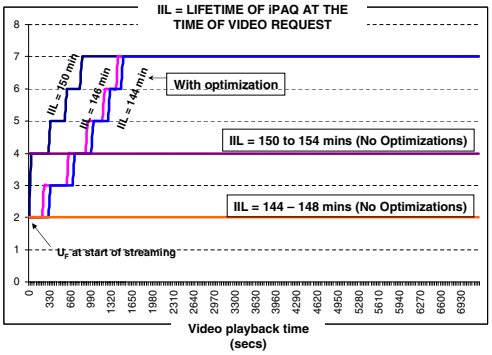

Figure 4. Utility Factor over time

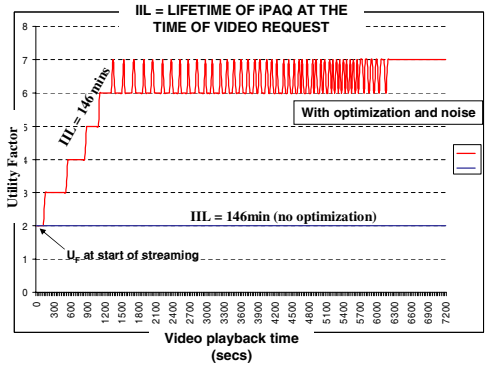

Figure 5. Utility Factor over time (with noise)

3. Stream Annotations for Power-Performance Tradeoffs

Our framework should be able to dynamically adapt to global changes in the system: for example, changes in the data stream trigger the local middleware runtime on the device to automatically adjust the architecture level parameters (like CPU frequency, display backlight, etc). Recent studies [12] have shown that multimedia workloads are characterized by quasi regular patterns in their execution, mainly because media encoding/decoding is composed of processing filters, applied in a specific order. The only changes are introduced by variations in the input data and the algorithm itself. Knowledge of data characteristics can be exploited at all levels in a multimedia streaming application (hardware, OS, middleware/network, application) and is especially important for portable devices where battery life and thus mobility are of utmost importance.

Focusing on the OS/hardware level, we analyze the stream of data during playback and annotate it with a summary of the information collected; this information will be used later at run-time for data-aware optimizations. Annotations typically capture patterns or trends in the data that are difficult/impossible or too time-consuming to gather at run-time on the handheld device and that can be exploited later for either power or performance benefits.

3.1. Annotation-Based DVS for Multimedia Playback

The processing unit (CPU) is one of the major consumers of power during multimedia playing, mostly due to the computationally intensive decoding of MPEG compressed video. On the other hand, MPEG compression is a relative regular sequence of multimedia kernels applied to the input stream. Based on the actual content of each MPEG frame, we predict the CPU load for decoding video frames and automatically adjust the DVS settings for better energy efficiency.

At the server side, we profile a representative set of video clips from the domain and apply regression fitting algorithms to find a function between frame sizes/distribution and decoding times. Next, we devise an estimation heuristic for frame decoding time and use it to control a DVS scheme at the client side. Our technique slows down the processor (saving power) during each frame decode to the lowest frequency that still allows it to finish before the frame deadline.

To evaluate the power savings, we compare our approach to the original case, where no DVS technique is in effect and the processor runs at full power all the time. We also compare our technique against a simple heuristic, which we call "simple WCET DVS". This heuristic assumes a constant processor frequency for the duration of the entire clip, chosen so that all frames can be decoded before the deadline (slows down the processor such that the worst case decoding time is still before the deadline). The "simple WCET heuristic" is similar to what a current DVS-capable device would actually perform and does not take advantage of the time difference between decoding different frames.

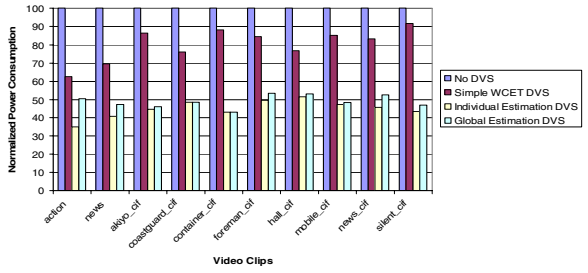

Figure 6. Power Savings Results

In our approach, we use the prediction algorithm and profiling individually on each clip (i.e. each clip is first profiled, then prediction used for DVS). The results are presented in Fig. 6. As we can see, the power savings are up to 50% over no DVS and up to 40% over the "simple DVS" (worst case

assumption for decoding times). Both these results refer to CPU power consumption.

3.2. Backlight Scaling Using Annotations

In mobile devices backlight dominates other components, with about 20-30% of total power consumption. On the other hand, power consumption of LCD displays increases with the backlight level intensity, yielding important savings for the entire system can be achieved if the LCD backlight is properly adjusted during the playback of a movie.

On a typical PDA screen, the perceived intensity of pixels is given by the formula: $I = \rho \times L \times Y$, where ρ is the transmittance of the LCD panel, while L and Y are the luminance of backlight and displayed image respectively.

During video playback, the entire range of luminance is not always completely used: for example a large number of scenes in a movie have darker scenes, i.e. scenes in which there are only a few or no pixels in the high luminance range. This allows us to increase the brightness of the image while simultaneously dimming the backlight for reduced power usage. We use annotations for storing luminance information for different scenes in a video stream. For reducing the power consumption during playback, we dim the backlight while at the same time compensating by increasing the luminance of the displayed image.

For each type of PDA we characterize the display and backlight performance (power vs luminance intensity vs backlight level). We perform this by displaying images of different solid gray levels on the handheld and capturing snapshots of the screen with a digital camera. For example, we noticed that for our handheld, while luminance is almost linear with the level of white in the image (gray level), it is not linear with the backlight level This allows us to easily compute the new backlight level needed to achieve a particular luminance level. In our experiments we also noticed that the power consumption of the LCD is almost proportional to the luminance.

We validate our results by using a digital camera and taking snapshots of the PDA displaying initial frame (reference screen snapshot) and comparing it with a picture of the same frames after backlight/brightness adjustment (compensated screen snapshot).

During profiling (at the media server), our technique uses a simple heuristic[10] to find and tag the various scene in a movie, where the maximum luminance level does not vary significantly. Then, for each scene the required level of backlight is computed and annotated to the video stream. Depending on the level of quality requested by the user, if it allows the compensation algorithm goes even further and trades off quality for power and an increase in battery life. We noticed that a very small number of pixels amount for the highest luminance levels. Therefore, we can safely remove a large number of these pixels without noticeable quality loss on the image displayed on the PDA, but with important savings for backlight during playback. The LCD backlight level is controlled by the OS level, based on the annotations found in the media stream (for each separate scene).

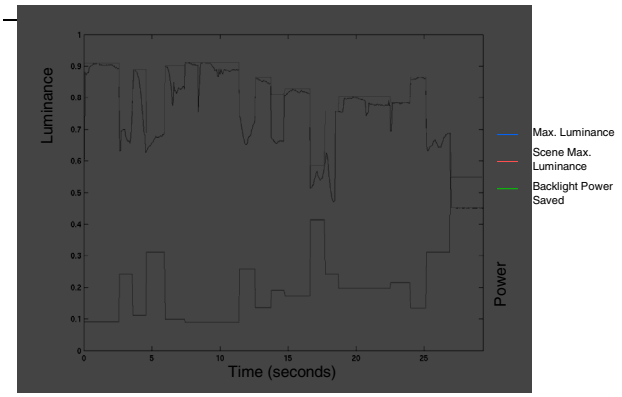

Figure 7. Backlight Compensation

Figure 7 presents our backlight adjusting technique results during a short video clip. It shows the original maximum luminance and our scene-grouping based on luminance level. It also plots the instantaneous power savings for the LCD backlight during playback.

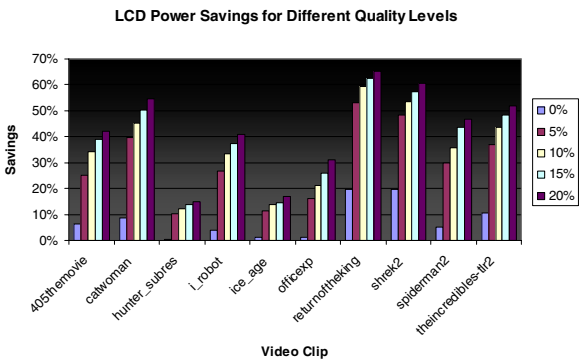

Figure 8. LCD Power Savings

On the client device, the user chooses a desired level of quality, which translates in a maximum percentage of pixel degradation allowed. We experimented with a number of different quality levels (0% to 20% quality loss). Even at the 5% quality lost we already start seeing a huge improvement in the backlight power consumption, and visual degradation is kept to a minimum (hardly noticed). The results (figure 8) show that up to 50% of the backlight power consumption can be saved using our approach, or even more if the user allows a more aggressive QoS-energy trade-off.

4. Parameter Optimization for Cross Layer Adaptation

At the application level, our framework optimizes various parameters available in the multimedia workload so that all given constraints are satisfied such as image quality, network bandwidth and power budget. The approach follows four steps: specification, profiling, optimization and code generation.

4.1. System Specification for Video Encoding Algorithm

The block diagram of a conventional video encoder like an H.263 encoder is shown in figure 9. The transform coding includes the following major steps: Motion Estimation (ME), Discrete Cosine Transform (DCT), Quantization (Q) and Variable Length Coding (VLC).

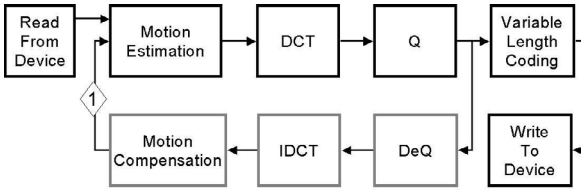

Figure 9. H.263 Encoder Algorithm

As seen from the above figure, we can adjust and optimize several application parameters to meet constraints such as power budget, image quality, network bandwidth, device computation ability and device memory size. For instance, we can increase quantization level to reduce the bitstream size for meeting bandwidth constraints, albeit at a loss of video quality. In order to study the QoS tradeoffs required to meet dynamic system constraints, we study the following components of the video encoding algorithm: implementation selection, algorithm parameters and system setting as shown in figure 10.

We implement several motion estimation algorithms: full search, diamond search [20], three step search [14], two dimensional logarithmic search [13], one at a time search [18] and so on. Similarly, we change the implementations of DCT, the quantization algorithm, and variable length coding block. One of our goals is to identify the relative tradeoffs achievable through each of these algorithms.

Multimedia applications allow us to tune the algorithm parameters. For instance, quantization level is defined in the H.263 encoding standard to control image quality and compressed data size. In addition, I:P frame ratio, frame rate and frame size can be chosen to meet image quality and compressed frame size. If we increase I:P frame ratio, we can decrease the compressed size with more computation for motion estimation since I frame does not require any motion estimation while P frame does.

The system setting is an important factor to affect overall system performance. Using an accurate profile of application behavior (e.g. execution profile, bandwidth usage), we can employ our cross-layer framework to perform better adaptations at the various system levels discussed earlier. For example, we can provide hints to the OS better scheduling decisions (e.g. DVS) as well as improve upon our proxy based remote adaptations.

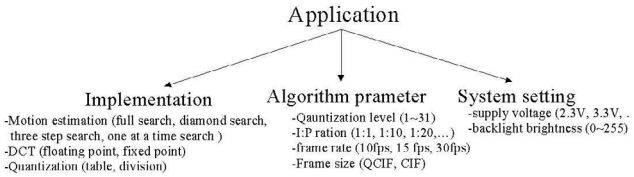

Figure 10. Adjustable parameters

4.2. Profiling

For profiling, we observe system performances by simulating or running an application on a real hardware platform. We measure image quality, compressed bit size, execution time and energy consumption by changing motion estimation algorithm, quantization value, and I:P ratio. We use diamond search with a quantization value of 5, 1:10 for I:P ratio. We take three standard video sequences for our experiments: CLAIRE.QCIF (less movement), GARDEN.QCIF (high movement) and FOREMAN.QCIF (a mix of both types of movements). The encoder algorithms are run on a Sharp Zaurus PDA platform consisting of an ARM processor (200 MHz) which runs Linux as operating system.

Energy, power and execution time are measured for each of the above mentioned algorithms. The peak signal to noise ratio (PSNR) is measured to get an estimate of the quality level achieved after encoding. The bit rate (number of bytes per frame in the encoded video) is also calculated to get an idea of the compression achieved by encoding.

Table 1 represents tradeoffs of image quality, execution time (power consumption) and bit rate between motion estimation algorithm. We use full search, diamond search, three step search, one at a time search, two dimensional logarithmic search, motion estimation without half-pel search and no motion estimation. Table 2 represents profiling results at various quantization values while Table 3 indicates results at I:P ratio.

ME	PSNR (dB)	Bitrate (bytes/frame)	Time (sec)	Energy (mJ)
FS	36.82	1758.84	159.34	141.67
DS	36.86	1848.93	85.11	74.94
TSS	36.80	1994.95	88.22	78.33
OTS	36.88	2345.12	75.31	75.67
TDS	36.85	1875.09	79.26	79.35
w/o HP	36.30	2202.65	64.94	54.56
NO	37.85	5489.09	65.25	59.15

Table 1. Tradeoff between motion estimation

Q	PSNR (dB)	Bitrate (bytes/frame)	Time (sec)	Energy (mJ)
1	46.81	11908.11	96.13	84.59
4	38.08	2336.17	79.12	69.63
8	33.76	1018.26	74.56	64.73
12	31.48	626.08	72.60	63.89
20	28.87	360.12	70.56	61.80
31	26.93	250.10	69.43	61.10

Table 2. Tradeoff between quantization

4.3. Experimental Results

To determine appropriate encoding parameters of our video application (combination of quantization, I:P ratio and motion estimation), we find pareto optimal points from the populated data. We then minimize execution time while satisfying minimum image quality and bandwidth limitations. The problem is formulated as an ILP and solved using an integer linear programming solver [2]. The solving time for integer linear programming is no more than 0.2 second on a P4 3.0GHz PC.

Figure 11 shows results with 2K bytes/frame for FOREMAN.QCIF (300 frames). The x axis shows constrained PSNR value to be met and the y axis is execution time. The "Q only" varies quantization value only without changing motion estimation and the I:P ratio. The "Q,ME" adapts both quantization value and motion estimation algorithm. If we consider more parameters then we can get shorter execution time which is proportional to total energy consumption. On an average, we expect that we can reduce execution time by 40% and 20% by changing the quantization step, motion estimation algorithm and I:P ratio than by adapting quantization step only, and by adjusting quantization step and motion estimation algorithm.

I:P = 1:x	PSNR (dB)	Bitrate (bytes/frame)	Time (sec)	Energy (mJ)
4	36.89	2378.78	73.12	73.98
8	36.64	1934.61	82.05	87.75
16	36.48	1707.65	87.06	95.39
64	36.29	1547.34	88.26	97.23
128	36.27	1531.59	89.64	99.69

Table 3. Tradeoff between I:P ratio

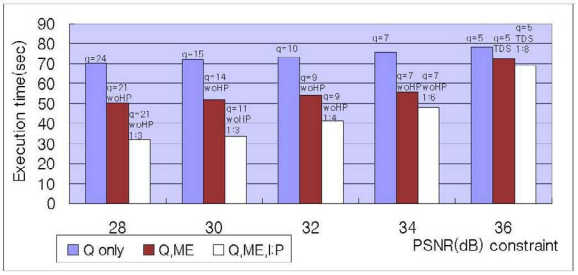

Figure 11. Minimization of execution time (bytes/frame $\leq$ 2000)

Finally, we generate a code with these parameter values computed above. The code chooses the proper value for each parameter from the pre-computed values at the previous step. Note that for a static environment (e.g., fixed network b/w, image quality), some parameters can be fixed; otherwise they should be chosen at run-time. In the measurement results, we reduce the execution time by 33% and 15% which are similar to the results in Figure 11 when the bytes/frame is 2000. When the bytes/frame constraint is 1000, the execution time is reduced by 30% and 10% than Q only and Q,ME respectively while when the constraint is 500, it is 17% and 20%. These results show our ability to select encoding parameter settings while minimizing execution time.

5. Related Work

Several research results have focused on analyzing the input data stream and deriving various techniques for improving either communication or computation in streaming applications; we review related efforts below.

The Aspire research group studies various data-shaping algorithms for mobile multimedia communication. They profile and annotate still images for improving transmission over a wireless channel usage (bandwidth, latency). In [11] the image data is compressed according to dynamic conditions and requirements. Content adaptation is classified depending on time (static, dynamic), content (to determine optimal compression) and goals of technique or metrics (constrained bandwidth, display size, response time).

Chandra and Ellis perform an informed quality-aware transcoding in [4], based on image characteristics. They find that a change in JPEG quality factor (compression metric controlled by quantization steps) directly corresponds to information quality lost. A prediction for computational overhead is applied, which approximates number of basic compu-

tation blocks based on image size, color depth and can predict output size for a particular transcoding.

In [5], the authors analyze the characteristics of images available on web sites (distribution of gif or jpg images, size, colors and quality). They clasify images in: bullets, lines, icons, banners, trueimages based on heuristics and analyze various transcoding techniques for reducind image size (reducing spatial geometry or thumbnailing, reducing the number of unique colors, changing the image format or compression). Tripathi and Claypool study different ways to reduce bandwidth in network transmission in [19], by either temporal scaling (dropping frames), quality scaling (reducing quality of frames) or spatial scaling (changing the size of frames). The quality degradation is evaluated through an user study.

Reducing backlight intensity as a means of saving power was also studied by Cheng et al. in [7]. Here the authors present a hardware based implementation for concurrent brightness and contrast scaling in a TFT-LCD display. The technique yields good power savings for still images, but it is not applicable to video sequences due of its frame based analysis. An application of backlight power optimization for streaming video applications is presented in [16]. The adaptation is coordinated by a middleware layer running on both the client and a intermediary proxy node.

[8] describes a number techniques for low-power TFT LCD display, one of which applies backlight luminance dimming with appropriate brightness and contrast compensation. These techniques result in good aggregated results for a number of different applications. Backlight compensation is further implemented in hardware and further studied for typical PDA applications in [6].

Unlike our efforts in the FORGE project, none of these related efforts perform a cross-layer approach for power-performance optimization.

6. Conclusion

In this paper, we discussed the various initiatives under the FORGE framework for studying energy and performance tradeoffs for mobile multimedia systems. By coordinating energy saving techniques at all system levels, the FORGE framework is able to improve user experience on mobile devices. In addition, for distributed applications the knowledge of the global system state such as network noise and mobility information can lead to better energy aware adaptations on a device. We show that exchange of global/local information is beneficial for all layers (OS, hardware, middleware, application) and the FORGE framework integrates a distributed middleware to perform remote optimizations and collect global system state. Our results indicate that such a cross-layer approach can result in better application performance and better energy utilization at the device. As an extension of our current efforts, we plan to develop adaptations involving multiple proxies and devices.

References

[1] http://dynamo.ics.uci.edu.

[2] http://www.cplex.com.

[3] http://www.redhat.com/docs/manuals/linux/RHL-9-Manual/ref-guide/ch-proc.html.

[4] S. Chandra and C. S. Ellis. JPEG compression metric as a quality-aware image transcoding. In *USENIX*, 1999.

[5] S. Chandra, A. Gehani, C. S. Ellis, and A. Vahdat. Transcoding characteristics of web images. In *MMCN*, 2001.

[6] N. Chang, I. Choi, and H. Shim. Dls: Dynamic backlight luminance scaling of liquid crystal display. In *TVLSI*, volume 12, pages 837–846, 2004.

[7] W.-C. Cheng, Y. Hou, and M. Pedram. Power minimization in a backlit tft-lcd display by concurrent brightness and contrast scaling. In *DATE*, page 10252. IEEE Computer Society, 2004.

[8] I. Choi, H. Shim, and N. Chang. Low-power color tft lcd display for hand-held embedded systems. In *ISLPED*, pages 112–117. ACM Press, 2002.

[9] R. Cornea, N. Dutt, R. Gupta, I. Krueger, A. Nicolau, D. Schmidt, and S. Shukla. FORGE: A framework for optimization of distributed embedded systems software. In *IPDPS*, April 2003.

[10] R. Cornea, A. Nicolau, and N. Dutt. Using annotations to facilitate power vs quality trade-offs in streaming appl ications. Technical report, University of California, Irvine, 2005.

[11] D.G.Lee, D.Panigrahi, and S.Dey. Network-aware image data shaping for low-latency and energy-efficient data services over the Palm wireless network. In *World Wireless Congress (3G Wireless)*, 2003.

[12] C. J. Hughes, P. Kaul, S. V. Adve, R. Jain, C. Park, and J. Srinivasan. Variability in the execution of multimedia applications and implications for architecture. In *ICCA*, 2001.

[13] J. R. Jain and A. K. Jain. Displacement measurement and its application in interframe image coding. *IEEE Trans- Commun.*, COM-29:1799–1808, December 1981.

[14] T. Koga, K. Iinuma, A. Hirano, Y. Iijima, and T. Ishguro. Motion compensated interframe coding for video conferencing. *In Proc. Nat. Telecommun. Conf.*, December 1981.

[15] S. Mohapatra, R. Cornea, N. Dutt, A. Nicolau, and N. Venkatasubramanian. Integrated power management for video streaming to mobile handheld devices. In *ACMMM*, November 2003.

[16] S. Pasricha, S. Mohapatra, M. Luthra, N. Dutt, and N. Venkatasubramanian. Reducing backlight power consumption for streaming video applications on mobile handheld devices. In *ESTIMedia, CODES-ISSS*, 2003.

[17] V. Raghunathan, C. Pereira, and et. al. Energy Aware Wireless Systems with Adaptive Power-Fidelity Tradeoffs. *VLSI Systems*, 2004.

[18] R. Srinivasan and K. R. Rao. Predictive coding based on efficient motion estimation. *IEEE Trans- Commun.*, COM-33:888–895, August 1985.

[19] A. Tripathi and M. Claypool. Improving multimedia streaming with content-aware video scaling. Technical Report WPI-CS-TR-01-02, Worcester Polytechnic Institute, 2001.

[20] S. Zhu and K. K. Ma. A new diamond search algorithm for fast block matching motion estimation. *ICICSP*, 1, 1997.