# Annotation Integration and Trade-off Analysis for Multimedia Applications

Radu Cornea, Alex Nicolau, Nikil Dutt

Donald Bren School of Information and Computer Science
University of California, Irvine, CA 92697-3425
{radu,nicolau,dutt}@ics.uci.edu

## Abstract

*Multimedia applications for mobile devices, such as video/audio streaming, process streams of incoming data in a regular, predictable way. Content-aware optimizations through annotations allow us to highly improve the power savings at the various levels of abstraction: hardware/OS, network, application. However, in a typical system there is a continuous interaction between the components of the system at all levels, which requires a careful analysis of the combined effect of the aforementioned techniques. We investigate such an interaction and we describe metrics for estimating the effect various trade-off have on power and quality. By applying our metrics at the various abstraction levels we show how better energy savings can be achieved with lower quality degradations, through power-quality trade-offs and cross-layer interaction.*

## 1. Introduction

Rapid advances in processor technology and the adoption of wireless communication have caused a shift in the computing industry towards mobile handheld devices like handhelds, PDAs and cellphones. At the same time, we find that these devices are increasingly being used in multimedia applications, common examples being on-demand content delivery (movie streaming) over the network and video conferencing. The main drawback of these technologies is the limited battery life associated with wireless enabled mobile devices.

Battery technology does not match the afore mentioned advances, remaining a limiting factor of operating life in portable devices. The main power consuming components of a handheld device are the *CPU*, the *display* and the *network interface*, each consuming around one third of the overall power.

In previous publications we have outlined techniques for the application of annotations at various levels of abstraction: hardware/OS [4], network [2], application [3]. However, in a typical system there is a continuous interaction between the components of the system at all levels, which requires a careful analysis of the combined effect of the aforementioned techniques. In this paper, we try to analyze the effect of combining techniques at different levels of abstraction, for different components of the system. We look at this problem with both no quality trade-offs and in the presence of power-quality trade-offs for even higher savings, with minimal quality degradation.

The paper is organized as follows: we start by integrating the power savings techniques at all levels and discuss the overall system savings. Then, we describe the metrics used for estimating the effect various trade-off have on power and quality. We apply the metrics at some of the abstraction levels and show how a better quality-power balance can be achieved through cross-layer interaction.

## 2. Data annotation

This section briefly discusses the concept of annotations. Data annotation analyzes the content of a data stream and annotates the collected information to the stream itself. Annotations typically capture patterns or trends in the data stream that are difficult/impossible or too time-consuming to gather at run-time on the mobile device and that can be later exploited for either power or performance benefits.

The advantage of annotating the data off-line is two-fold. First, there is no overhead for doing all the work at runtime, by the client device. Second, because the information is known in advance, more optimizations are possible (for example the network optimizations).

## 3. System architecture

The system model assumed is depicted in Figure 1. It includes a multimedia server, users with low-power wireless devices and other network equipment along the way. The multimedia servers store media content and stream it to clients upon requests issued by the users.

1-4244-0910-1/07/$20.00 

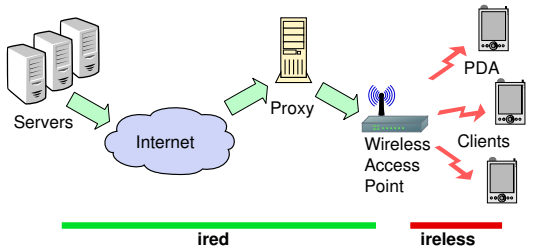
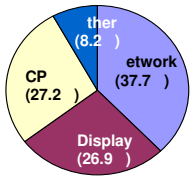



**Figure 1. System framework**




**nergy Distribution**
**( efore ptimi ation)**

**Figure 2. Power distribution for a typical PDA**

The communication between the client device and the servers can be routed through an optional proxy server – a high-end machine that has the ability to process the video stream in real-time.

Annotations can be generated and added to the video stream at either the server side or the proxy node, therefore saving the client of any additional work.

## 4. Annotation Integration

The effect of applying individual annotation techniques at each abstraction level can be better understood in the scope of the whole system. Each component of the system (e.g. CPU/memory, network card, LCD display) may be optimized for power through annotations, but the overall effect can only be estimated by integrating all the optimizations.

The techniques we applied at the hardware, network and application levels are practically independent in our case. Each technique only affects one component in the system and the stream is not changed in any significant way. To evaluate the system level contribution of annotations we apply the three techniques presented in previous publications:

- **hardware**: annotation-based frame decoding, for CPU/memory power optimization [4]
- **network**: burst-transmission based on annotations [2]
- **application**: backlight scaling for display power reduction [3]

In our evaluation, we only allow minimal quality degradation for the backlight scaling and a loss-less application for the other two techniques.

Because the techniques are practically independent, savings at one level do not affect savings at other levels, in other words savings are additive. If that was not the case, we would have to take into account the interaction between the application of techniques that interfere between themselves.

In order to measure the system level savings, we chose a number of video clips from the multimedia community, ranging from very still to very dynamic. For the PDA simulated, we assume the following power distribution (Figure 2): CPU 27.2%, network card 37.7%, display 26.9% and other components 8.3%. The power numbers correspond to an iPaq 5555 PDA.

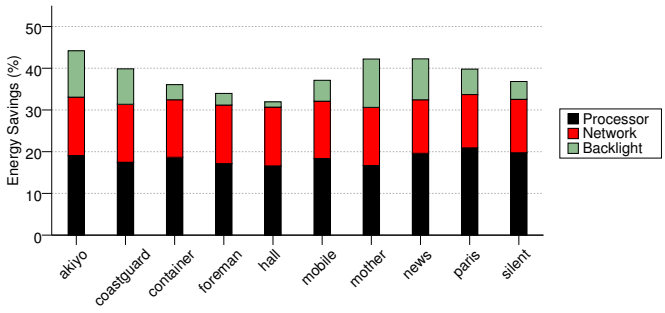



**Figure 3. Total power savings results (CIF videos)**

The results show an average of 37% power savings for the entire system for the video clips selected (Figure 3). What we can observe from the graph is that network power savings tend to be constant, due to the similar bitrate encoding of the videos. Backlight contribution is smaller for videos with bright backgrounds (foreman, hall, container) and larger for darker video clips (mother, akiyo). CPU contributions is less in the case of video with more action (coastguard, hall) and more for the predominantly static videos (paris, silent).

The conclusion we can draw from analyzing these videos is that the amounts and relative contribution of each component of the system to the total power savings can vary immensely, being influenced by the factors that affect the application of our techniques (e.g. amount of highlights, motion, bitrate). To achieve better savings, we should target each video individually and apply the relevant power savings for it.

## 5. Power-Quality Tradeoffs

When compared with general purpose applications, multimedia applications have special advantages. The compression used in video and audio stream is typically lossy, which means that some quality degradation occurs even from the compression step. Moreover, multimedia has soft real-time

requirements. This means that frames or part of a frame can be lost without important consequences.

The lossy compression and soft real-time requirements of multimedia applications allow specific quality of service trade-offs, which are not available for general purpose applications. When trading-off quality for power, energy savings can be substantial with minimal quality loss, if the trade-off is done with knowledge of the characteristics of the video stream.

The challenge is finding a good objective quality assessment metric, which is able to predict (estimate) the quality degradation for the various power saving techniques that can be combined with quality loss.

### 5.1. Quality Assessment

Since at the end of the transmission flow are the users (humans), the goal of any quality assessment metric is to develop quantitative metrics that can automatically predict as accurately as possible the perceived image quality. Humans are subjective by nature, therefore most of these metrics are evaluated against a mean opinion score (MOS), which is computed by averaging answers from a large number of observers.

Traditionally, there are a few metrics very popular for image processing: MSE and PSNR. Assuming $x_i$ and $y_i$ are the luminance of pixels belonging to two different images and $N$ is the total number of pixels in each image, the *mean squared error* (MSE) is defined as:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(x_i - y_i)^2$$

Often, a logarithm scale is used to better account for the human perception and the metric used is the *peak signal-to-noise ration* (PSNR):

$$PSNR = 10log_{10}\frac{L^2}{MSE}$$

$L$ is the dynamic range of pixel intensities (for example, 255 for 8-bit gray-scale images).

However, both these metrics compare the images at a pixel-level, without taking account of any positional information in the image or the characteristics of the human eye, which tends to average pixels instead of seeing each pixel separately. Therefore, both MSE and PSNR have a poor correlation to the perceived image quality.

There are a number of alternatives to MSE and PSNR. Wang et al present in [6] a new metric called Structural SIMilarity (SSIM) Image Quality Index:

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}$$

where

$$\mu_x = \frac{1}{N}\sum_{i=1}^{N}x_i, \qquad \mu_y = \frac{1}{N}\sum_{i=1}^{N}y_i,$$

$$\sigma_x^2 = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2, \qquad \sigma_y^2 = \frac{1}{N-1}\sum_{i=1}^{N}(y_i - \bar{y})^2,$$

$$\sigma_{xy} = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})$$

$x_i$ and $y_i$ are pixel values, while $N$ is the total number of pixels.

The quality index for the entire image, mean SSIM, is computed through averaging:

$$MSSIM(X,Y) = \frac{1}{M}\sum_{j=1}^{M}SSIM(x_j, y_j)$$

This metric is shown to provide very good correlation with a human generated mean opinion score (MOS). In subsequent experiments, we will use this image quality metric (MSSIM, or shortly, SSIM) for quality assessment.

### 5.2. Quality Assessment for Videos

At the lowest level, videos are just a sequence of images (frames), therefore we are estimating the quality degradation by applying the SSIM metric to each individual frame and averaging the result. We show below what this means for different techniques:

- backlight scaling: in this case, frame compensation will produce saturated (clipped) pixels in frames, so by computing the SSIM index between the original and compensated frame we are accounting for the image degradation. The more clipped pixels, the lower the quality of the video clip. The quality index for the entire stream is estimated by averaging the quality index of each individual frame in the clip.

- frame decoding: this technique is more challenging, because it works in the temporal domain (B frame dropping). However, we can still estimate the quality of the video using SSIM in a simple way: when frame are lost, usually video players display the previous frame. For example, if the initial frame sequence was ABCD and frame C was lost, the sequence displayed would be ABBD, therefore the degradation is given by the difference between B and C. This way, we are also accounting for the jerkiness in the video, caused by frame loss: if the video was mostly static, the difference between B and C was minimal and the lost frame was barely visible; on the other hand, if the difference between B and C is high, the SSIM difference would be larger to account for the jerky motion in the video. The quality index for the entire video is computed as before, through averaging.

### 5.3. Quality Composition

In the previous sections we explained how to estimate the quality degradation (or quality index) for the application of a single degrading technique, applied on one component of the system. The problem becomes more difficult in the presence of multiple techniques, each applied to one abstraction level and each degrading the quality of the stream in a different way, for power savings. In the most general case, there may be cross-interference between these techniques, and therefore the overall quality degradation may become hard to compute.

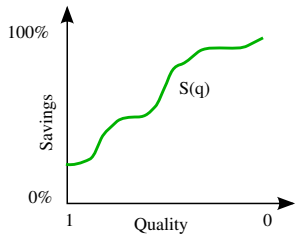



**Figure 4. Power savings variation with quality**

In the simplest case, when the techniques are independent and applied on different components of the system, overall quality degradation and power savings are easier to estimate. Let us assume a system where two power optimization techniques $A$ and $B$ are applied on a stream of multimedia content of initial quality $q_i$. The quality of the stream after applying both techniques on the stream is the output quality of the system $q_o$. Now, if we denote by $S_A(q_a)$ and $S_B(q_b)$ the functions describing the variation of power savings with quality index (as in Figure 4), then the system can be represented as in Figure 5.

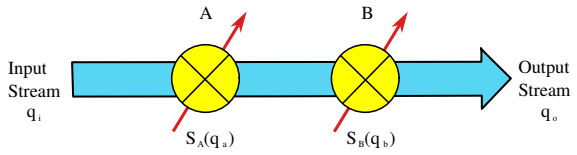



**Figure 5. Quality and power composition for multiple techniques**

Similar to the factors in the product computing the SSIM metric (each measuring one type of degradation), the quality degradations are applied here one after the other, "on the same stream", so their effect is multiplicative (each contributes as a factor in the final quality). In other words the overall quality degradation is:

$Q = q_a * q_b, \qquad q_o = q_i * Q$

where all $q$ factors represent values of the SSIM index for the corresponding technique and $0 <= q <= 1$.

On the other hand, the power savings apply to different parts of the system (CPU, network, display), so their effect is additive. The total power savings can be computed with the formula:

$S = S_A(q_a) + S_B(q_b)$

where $S_A$ and $S_B$ are contributions of $A$ and $B$ to total power savings for the entire system.

In a more general case with $n$ different techniques applied on different parts of the system:

$Q = q_1 * q_2 * \cdots * q_n$

$S = S_1(q_1) + S_2(q_2) + \cdots + S_n(q_n)$

### 5.4. Cross-layer Trade-off Analysis

The problem of finding the best combination of power-quality trade-offs becomes an optimization problem. For example, in the case of the previous two techniques $A$ and $B$, the problem becomes maximizing power savings savings $S = S_A(q_a) + S_B(q_b)$, while maintaining the quality of the stream $Q = q_a * q_b$ high (low degradation happens when $q$ is as close as possible to 1).

However, in real world user may prefer some types of degradation to others (e.g. dropping frames as opposed to degrading image content). In that case, we can give weights for the importance of each quality degradation factor to the overall product. The formula for the total quality degradation becomes:

$Q = q_a^{\alpha} * q_b^{\beta}$ ,

where $\alpha$ and $\beta$ specify the relative importance of the techniques $A$ and $B$, as specified by the user. If all techniques have equal importance, $\alpha = \beta = 1$.

By introducing the relative importance for each technique, our general formulas become:

$Q = q_1^{p_1} * q_2^{p_2} * \cdots * q_n^{p_n}$

$S = S_1(q_1) + S_2(q_2) + \cdots + S_n(q_n)$

where $p_1, p_2, \ldots, p_n$ represent the relative importance of the corresponding technique.

## 6. Experimental Results for Power-Quality Trade-off

In this section we are applying the power-quality trade-off theory to real-world situations. First, we are evaluating the trade-offs between power and quality for a single component. Then, we are combining more components of the system and show how a cross-layer interaction (trade-off) could yield higher savings, with lower quality degradation.

### 6.1. Results for Single Layer

Our first experiment involves the LCD display backlight scaling and its power-quality trade-off. We evaluated a number of video clips from multimedia community, estimating the quality index and power savings for various levels of pixel loss (see [3]). The results are presented in Figure 6.

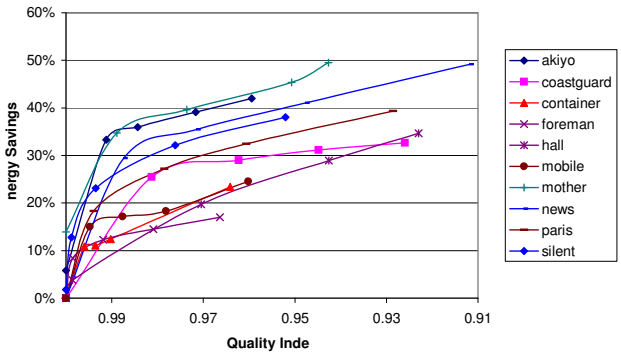



**Figure 6. Power-quality trade-off for the LCD**

We observe a large variation between the different clips. For video clips with bright backgrounds ('hal', 'container', 'foreman'), the power savings quickly saturate as we decrease the quality, and then only slightly increase. For darker videos on the other hand ('akiyo', 'mother') the savings are substantial (up to 35%) as we decrease the quality to 0.99, after which the savings increase slightly slower. In conclusion, the most beneficial region of the graph is the initial part when the savings grow almost linearly, when quality index is very high (close to 1). This is the region of the graph that yields the best power-quality trade-offs and should be selected in a real world implementation.

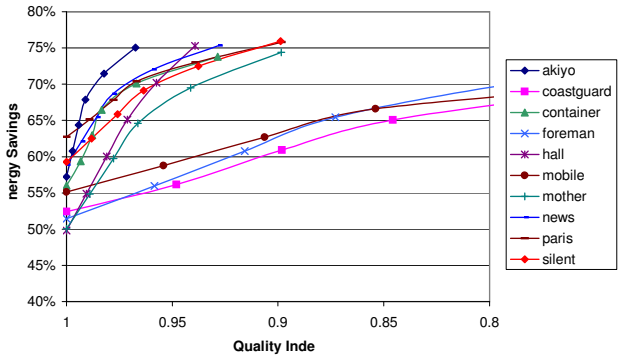



**Figure 7. Power-quality trade-off for the processor**

The second experiment was performed for our annotation-based DVS for frame decoding (see [4]). To allow quality vs power trade-offs, we allow dropping a percentage of the B frames. We choose to drop the B type frames, because they are not required to decode any other frames (unlike the I and P frames). We experimented with dropping between 0 and 100% (all) B frames. The quality degradation is computed using SSIM, as mentioned before. The computed graph is presented in Figure 7.

In contrast with the previous backlight scaling graph, here the savings start at a high value even for no quality loss. This is dues to the DVS algorithm , which is able to save power through a better estimation of frame decoding time. Video clips with a lot of motion ('foreman', 'mobile', 'coastguard'), tend to rapidly degrade the quality as we drop more B frames, with only a slight increase in savings. On the other hand, for mostly static clips ('paris', silent', 'container', 'akiyo'), the quality degradation is minimal even with all B frames missing and the power savings increase more rapidly with the degradation. Again, the most beneficial region is the one near very high quality (close to 1).

### 6.2. Results for Cross-layer

As we could see from previous section, for some video clips the power savings are minimal unless the quality of the video degrades considerably. This is a limitation due to the fact that we are performing the trade-off at a single level and do not take advantage of any possible cross-layer trade-offs.

If we investigate two or more levels at the same time, we can better balance the power and quality loss, depending on the variation between them for each level (as shown in previous chapter).

An example scenario is given below for the video clip 'coastguard'. As observed before, this clip contains a lot of motion and has smaller DVS power savings. If we draw the power-quality graphs for both backlight scaling and frame decoding, we can observe how each power-quality function varies with the amount of quality lost (Figure 8).

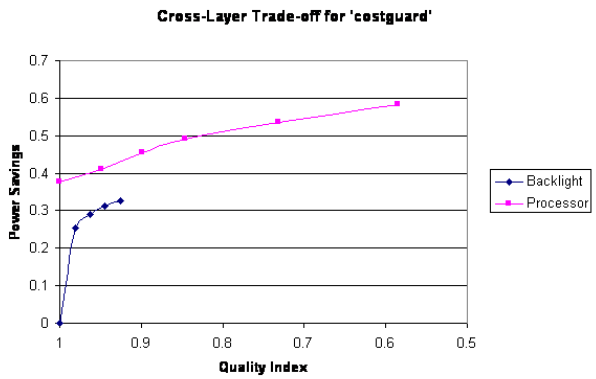



**Figure 8. Cross-layer trade-off for 'coastguard'**

From our previous discussion, the quality and savings for a combination of two independent techniques can be estimated using the following formulas:

$Q = q_a * q_b$

$S = S_A(q_a) + S_B(q_b)$

where $A$ is the backlight scaling technique and $B$ is DVS for frame decoding.

In our scenario, let us assume we are trying to find a good power quality trade-off that would degrade the image quality by $Q = 0.95$. These are two possible ways to reach the required overall quality:

- allow degradation from backlight scaling only: $q_a = 0.95, q_b = 1$. As a result, $Q = 0.95$ and total savings are $S = 0.3(0.31 + 0.38) = 23\%$ of total power. The $0.3$ factor is the relative contribution to total power of backlight (about one third).
- allow degradation from frame decoding only: $q_a = 1, q_b = 0.95$. In this case, $Q = 0.95$, but the total savings reduce to $S = 0.3(0 + 0.41) = 13.6\%$ of total power. We can observe that savings are reduced to almost half the previous ones, and this is due to the fact that we did not take advantage of the great backlight power savings in this region of the graph and instead we chose to use the smaller savings from DVS (frame decoding).

This example shows the large differences between various possible trade-offs and shows how important a cross-layer power-quality trade-off is for achieving the best power savings with the minimal quality degradation.

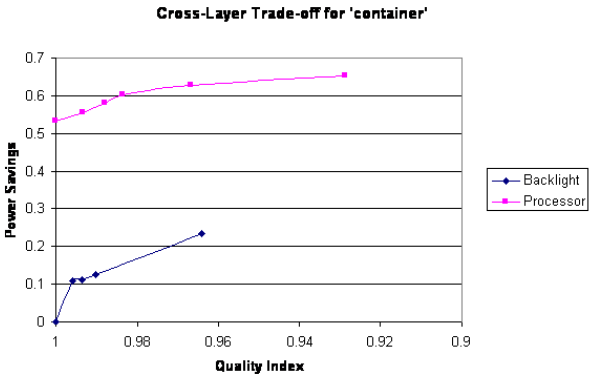



**Figure 9. Cross-layer trade-off for 'container'**

A second example scenario is presented in Figure 9 for the video clip 'container' (mostly static, with bright backgrounds). Here we can observe that power savings from backlight scaling are high until around $q = 0.996$ and minimal after that. On the other hand, due to the static nature of the video, processor savings (from frame decoding) increase steadily as the quality of the video decreases. In a cross-layer trade-off, the backlight savings would be preferred until quality drops to less than $q = 0.996$, after which savings from the processor are higher.

This example again shows how important a content based, cross-layer trade-off is for extracting the best power savings from a combination of system components, with the minimal quality degradation possible.

## 7. Related work

Other efforts to study data patterns include those of the Mesdat research group that studies various data-shaping for mobile multimedia communication. They profile and annotate still images for improving transmission over a wireless channel usage (bandwidth, latency). In [5] 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 performs an informed quality aware transcoding in [1], based on image characteristics. He finds 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 computation blocks based on image size, color depth and can predict output size for a particular transcoding.

The GRACE project [8] proposes 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 [7], a resource aware admission control and adaptation is suggested for multimedia applications for optimal CPU gains.

## References


[1] S. Chandra and C. S. Ellis. JPEG compression metric as a quality-aware image transcoding. In *USENIX Symposium on Internet Technologies and Systems*, 1999.

[2] R. Cornea, A. Nicolau, and N. Dutt. Annotation based multimedia streaming over wireless networks. In *Fourth IEEE Workshop on Embedded Systems for Real Time Multimedia*, October 2006.

[3] R. Cornea, A. Nicolau, and N. Dutt. Software annotations for power optimization on mobile devices. In *Design Automation and Test in Europe*, March 2006.

[4] R. Cornea, A. Nicolau, and N. Dutt. Video stream annotations for energy trade-offs in multimedia applications. In *International Symposium on Parallel and Distributed Computing*, July 2006.

[5] 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.

[6] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. . Simoncelli. Image quality assessment: from error visibility to structural similarit y. *Image Processing*, 13(4):600–612, 2004.

[7] W. Yuan and K. Nahrstedt. A Middleware Framework Coordinating Processor/Power Resource Manageme nt for Multimedia Applications. In *IEEE Globecom*, Nov 2001.

[8] W. Yuan, K. Nahrstedt, S. Adve, D. Jones, and R. Kravets. Design and Evaluation of a Cross-Layer Adaptation Framework for Mobil e Multimedia Systems. In *MMCN*, January 2003.