Power-Aware Multimedia Streaming in Heterogeneous Multi-User Environments*

Radu Cornea, Shivajit Mohapatra, Nikil Dutt, Alex Nicolau, Nalini Venkatasubramanian
Dept. of Information & Computer Science
University of California, Irvine, CA 92697-3425
{radu,mopy,dutt,nicolau,nalini}@ics.uci.edu

Abstract

Streaming multimedia content to heterogeneous handheld devices is a significant research challenge, due to the diverse computation capabilities and battery lifetimes of these devices. A unified framework that integrates low level architectural optimizations (CPU, memory), OS power-saving mechanisms (Dynamic Voltage Scaling) and adaptive middleware techniques (admission control, transcoding, network traffic regulation) can provide significant improvements in both the system performance and user experience. In this paper, we present such an integrated framework and investigate the trade-offs involved in serving distributed clients simultaneously, while maintaining acceptable QoS levels for each client. We show that the power savings attained at both CPU/memory and network levels can be aggregated for increased overall performance. Based on this, we demonstrate how an integrated framework, that supports tight coupling of inter-level parameters can enhance user experience on handheld devices.

1 Introduction

Advances in processor and wireless networking technology are generating a new class of multimedia applications (e.g. video streaming) for mobile handheld devices. Typically, these devices have limited resources - lower processing power, memory, display capabilities, storage and limited battery lifetime as compared to desktop/laptop systems. On the other hand, multimedia applications have higher Quality of Service(QoS) and processing requirements which tend to make them extremely resource-hungry. In addition, human perception of multimedia quality is significantly influenced by the device specific attributes (e.g form factor) [7]. Therefore, delivering high quality multimedia content to mobile handheld devices, while preserving their service lifetimes remain competing design requirements. This introduces key research challenges in the design of multimedia applications, intermediate adaptations and low-level architectural improvements of the device. Moreover, distributed environments where heterogeneous devices perform simultaneous streaming pose an important challenge for the system designer. The distribution poses new problems related to mobility, shared resources, and QoS trade-offs for accommodating an increased number of users.

Recent years have witnessed researchers aggressively trying to propose and optimize techniques for power and performance trade-offs for realtime applications. Different solutions have been proposed at various computational levels: architecture (hardware) – caches/memory optimizations [17], dynamic voltage scaling(DVS) [10], dynamic power management (DPM) of system components (disks, network interfaces), efficient compilers techniques – and application/middleware based adaptations [13]. However, an interesting disconnect is observed in the research initiatives undertaken at each level. Power optimization techniques developed at each computational level have remained seemingly independent of the other abstraction hierarchy levels, potentially missing opportunities for substantial improvements achievable through cross-level integration. The cumulative power gains observed by aggregating techniques at each stage can be potentially significant; however, it also requires a study of the trade-offs involved and the customizations required for unified operation. For example, decisive middleware/OS based adaptations are possible if low-level information(e.g optimal register file sizes & cache configurations) is made available; similarly low level architectural components (e.g CPU registers, caches etc.) can be optimized if the architecture is cognizant of higher-level details such as specific video encoding. The interaction between different layers is even more important in distributed applications where a combination of local and global information helps and improves the control decisions (power, performance and QoS trade-offs) made at runtime. Fig. 1 presents the different computation levels in a typical handheld computer and the cross layer interactions for optimal power and performance deliverance.

The purpose of our study is to develop and integrate hardware based architectural optimization techniques with high level operating system and middleware approaches (Fig. 1), for improvements in power savings and the overall

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

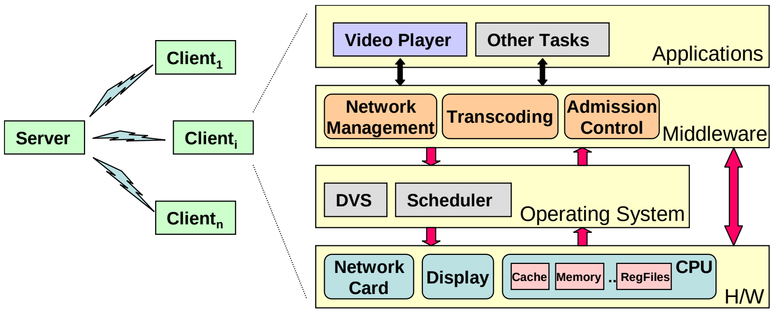

Fig. 1: Abstraction layers in a distributed multimedia streaming application

user experience, in the context of video streaming in a distributed environment of heterogeneous low-power handheld devices.

Multimedia applications heavily utilize the biggest power consumers in modern computers: the *CPU*, the *network* and the *display*. Therefore, we aggregate hardware and software techniques that induce power savings for these resources.

To address the challenge of integrating techniques at different levels (hardware, OS, middleware, user) we have adopted a multi-phase approach. First, low-level architectural tuning knobs are identified and combined with compilation techniques for optimized CPU performance; then, ideal operating points are determined for video streams of specific quality levels using these knobs; we then study the trade-offs involved in performing DVS along with our low-level optimization methods and evaluate the power gains of the wireless network interface using an adaptive middleware technique for a typical network with multiple users (noise). In this paper we start by summarizing these previous results. Next, the effect of multiple users on system resources is estimated, to drive our integrated framework. We then present a feedback-based middleware for multi-user power-aware admission control, quality and power-supported video transcoding; we study power vs. quality tradeoffs in the context of a distributed network of handheld computers. Finally, experiments with the integrated approach are presented, withe the final goal of improving the overall user experience (satisfaction) in the context of streaming video to handheld computers in a distributed environment.

By integrating the above techniques we are able to enhance the individual user experience and dynamically perform adaptation to also provide an improved overall system performance (among all users). Our previous results [7] have shown that architectural optimizations can give as much as 57% energy savings for the CPU and memory and middleware techniques up to 78% savings in the power consumption of the network interface card. Based on these combined results, our framework is able to accommodate a larger number of users in the system and serve clients with an increased level of video quality (even for devices that were unable to stream a video clip to completion initially, due to low battery life).

2 System Architecture

We assume the system model depicted in Fig. 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 (AP) and users with low-power wireless devices. The circles represent the noise at the access point due to network traffic introduced by "other" users. 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 is routed through the proxy server, that can transcode the video stream in real time.

The middleware executes on both the handheld devices and the proxy, and performs two important functions. On the devices, it obtains residual energy availability information from the underlying architecture and feeds it back to the proxy and relates the video stream parameters and network related control information to lower abstraction layers. On the proxy, it performs a feedback based multi-user power aware admission control and realtime transcoding of the video stream, based on the feedback from the device. It also regulates the video transmission over the network based on the noise level and the video stream quality. Additionally, the middleware exploits dynamic global state information(e.g mobility info, noise level etc.) available at the directory service and static device specific knowledge (architecture, OS, video quality levels) from the static rule base, to optimally perform its functions. The rate at which feedbacks are sent by the device is dictated by administrative policies like periodic feedback etc.. Moreover we assume that network connectivity is maintained at all times.

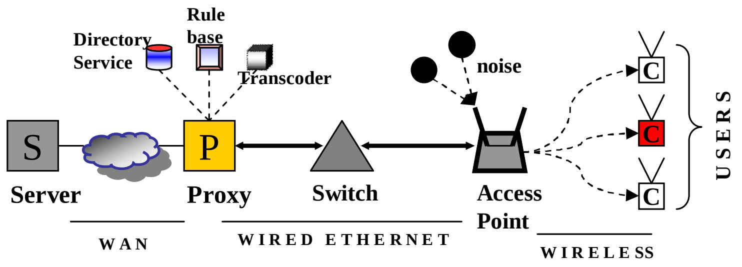

Fig. 2: System Model

Our goal is to provide an optimal user experience and maintain an acceptable utility factor for the system. We define an “acceptable utility factor” to be obtained when the system can stream the highest possible quality of video to the user such that time, acceptable quality and power constraints are satisfied (i.e the video clip runs to completion, at a quality level above or equal to the one the user specified). To accomplish this it is important to understand the notion of video quality for a handheld device and its implications on power consumption. Video applications introduce the notion of human perception of video quality as an important measure of performance. Moreover, user perception of quality is significantly influenced by the environment and the viewing device(e.g PDA) [1]. For example, most people are able to differentiate between close video quality levels on laptop/desktop systems; however only few are able to differentiate between close quality levels on a handheld device. It is hard to programmatically identify video quality parameters(a combination of *bit rate, frame rate and video resolution*) that produce a user perceptible change in video quality and/or a noticeable shift in power consumption.

In previous work [7] we established that users can only distinguish between mostly eight different levels of quality on a handheld device. Based on this, we selected eight quality levels, between Q1 (very high, like original) to Q8 (very low), chosen such that there is a perceptible video degradation and power variation between different adjacent levels. We identified transformation parameters (*bit rate, frame rate and video resolution*) for our proxy-based realtime transcoding, in order to generate the different quality level videos (transcoding is a process of decoding the original clip and re-encoding it at different parameters). Profiled average power consumption values were used to perform a high-level (coarse) power aware admission control for the system (this controls how users are accepted into the system and how resources are allocated to them in order to maintain a desired QoS). Similar device specific transformations can be made for other portable computers. More importantly, we also optimize our low-level architecture based on these discrete video quality levels. This approach provides us with two significant advantages: (i) Real time stream quality transformations can be performed with no overheads of dynamically determining quality degradation parameters, (ii) using the architectural tuning, optimized operating points can be pre-determined for a particular video stream quality. With the knowledge of the stream qualities, low-level cpu voltage scaling can also be improved. The following sections summarizes the architectural and middleware optimizations that are integrated into our system, followed by the system level optimizations and final integration.

3 Optimizations for Multimedia Streaming

Managing a distributed multimedia streaming system is a complex task and requires coordination at different levels of abstractions. We start by focusing on a single client model and identifying areas where power and performance optimizations can be performed, with the end result of an improved user experience. This section contains a summary of our previous work, in the context of the current endeavor. For more details on these techniques, see [7].

3.1 Hardware-level Adaptation

Hardware design techniques that identify multiple modes of operation and dynamic reconfiguration can be employed to attain high power and performance gains at the CPU architecture level. In order to optimize the architecture for delivering optimal energy performance, we first studied the low level functional units that have maximal impact on power consumption.

We identified ”knobs” for these components that can be made available to the higher abstraction levels for dynamically tuning the hardware for MPEG video applications.

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

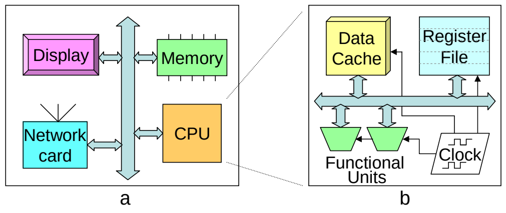

Fig. 3: Main Components of a Handheld Device

3.1.1 Quality-driven Cache Reconfiguration

Through extensive simulation, we identified the subcomponents of the CPU that consume the most power and observed the power distribution inside the CPU for MPEG decoding. It follows that internal CPU caches account for the largest part of the energy consumed by the processor. Moreover, the relative power contribution of the internal units of the CPU does not vary significantly with the nature or quality of the video played.

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

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 dependent on cache size and associativity. By profiling this function for the entire search space of available cache configurations, we determine the optimized operating point for that video quality.

We found out that for all video qualities such an 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.

We should mention that while current processors (including the ARM core on iPaq) in general do not exhibit such aggressive architectural reconfigurations, except for special purposes, there are many research projects on this and eventually the techniques will be incorporated into more future processors.

3.1.2 Integrated Dynamic Voltage Scaling

A different knob for controlling power drawn by the processor is dynamic voltage scaling (DVS) [10]. Voltage scaling is a method for trading-off processor speed against power, by lowering both voltage and operating frequency (power consumption when running at a high speed and voltage is much larger than when running at low speed and voltage). This technique provides significant savings for MPEG streaming as it allows tradeoffs for transforming the frame decoding slack time (CPU idle time) into important power savings.

In MPEG decoding, frames are processed in a fraction of the frame delay ($\boldsymbol{F_d = 1/frame\_rate}$). The actual value for the frame decoding time $\boldsymbol{D}$ depends on the type of MPEG frame being processed (**I** – intra-coded, **P** – predicted and **B** – bidirectionally predicted frames) and also on the cache configuration (size, associativity) and DVS setting (frequency, voltage). In our study, we assumed a buffered based decoding, where the decoded frames are placed in a temporary buffer and are only read when the frame is displayed. This allowed us to decouple the decoder from the displaying; decoding time was still different for different frames, but we could assume an average $\boldsymbol{D}$ for a particular video stream/quality. The difference between the average frame delay and actual frame decoding time gave us the slack time $\boldsymbol{\theta} = \boldsymbol{F_d} - \boldsymbol{D}$. When we performed DVS, we slowed down the CPU so as to decrease the slack time to a minimum, so we computed an operating frequency and voltage that minimized the slack $\boldsymbol{\theta}$.

Having the best DVS setting for each cache configuration and quality level, we looked at the effect of the integrated approach on the power consumption. The DVS is not totally independent of the cache reconfiguration technique (cache configurations with a largest slack time allow for higher DVS based power reductions) and as a result it effectively reshaped the total power consumption. Through simulation, we found the best operating points for the DVS/cache reconfiguration combined approach.

3.2 Architecture-Aware Middleware Adaptation

The gains obtained from the lower levels (architecture) can be further amplified if combined with techniques on higher levels (middleware). An adaptive middleware framework at the proxy can dynamically intercept and doctor a video stream to exactly match the video characteristics for which the target architecture has been optimized. Additionally, it can regulate the network traffic to induce maximal power savings in a network interface.

3.2.1 Network Traffic Regulation

We developed a proxy-based traffic regulation mechanism to reduce energy consumption by the device network interface. Our mechanism is able to dynamically adapt to changing conditions in the network and specific attributes of the wireless transmission (bandwidth, access point buffering). The packets are transmitted into optimized bursts of video by the proxy, along with control information. Since wireless network cards typically consume significantly more power in active vs sleep mode (about one order of magnitude [9]), our goal was to optimize the video burst sizes in order to maximize energy savings without performance costs.

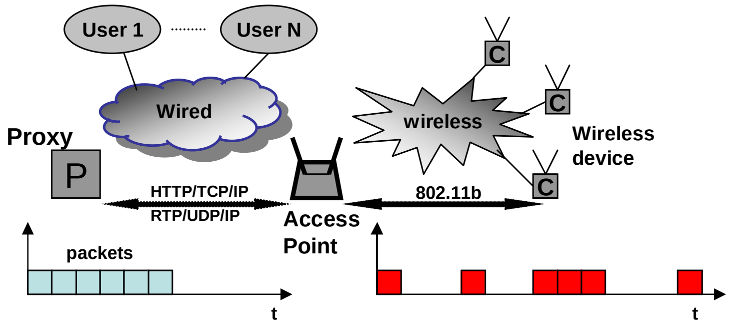

Fig. 4: Wireless Network

The analysis for the above power saving approach was performed using a realistic network framework(Fig. 4), in the presence of noise and AP limitations. The proxy middleware buffers the transcoded video and transmits several frames of video in a single burst along with the time for the next transmission as control information. The device then uses this control information to switch the interface between active/idle mode and sleep mode. A queueing theory analysis was used to predict packet loss rates at the access point due to limited buffering capacity, in the presence of multiple users (packets introduced by other users were modeled as exernal noise). We observed that large values of the burst length can result in packet losses at the access point and/or buffer overflows at the device. We acknowledged that a QoS aware preferential service algorithm at the access point can impact power management significantly. The above analysis could be used by an adaptive middleware to calculate an optimal burst length for any given video stream and noise level.

4 System-level Optimizations

In the previous sections we studied multimedia streaming/adaptation in a scenario where a single client was served by the proxy node. While this allowed us to concentrate on the server-proxy-client interaction, a real system would encompass multiple heterogeneous client devices being served simultaneously by one or more proxy nodes. Today's handheld devices exhibit a large variation in terms of multimedia playing capabilities. This includes processing power, display size, battery life, factors that in our approach drive multimedia adaptation on the proxy. In order to provide the best user experience at the device and be able to accommodate an increased number of clients at the same time, the proxy/middleware level must be cognizant of architecture level characteristics of each of the devices being served. This allows the transcoder on the proxy to better shape the streaming traffic to the devices so that a maximum number of clients can be served with a QoS above the requested threshold.

Extending our system architecture to multiple users unfolds new problems, like network congestion, proxy node resource exhaustion, etc. We address these problems and present possible solutions in the next sections.

4.1 Transcoding Multiple Streams

When performing transcoding for multiple users, the proxy node can soon become saturated, as this process is a very CPU intensive, especially if real-time is a requirement. *Transcoding* a video stream typically involves two steps (Fig. 5):

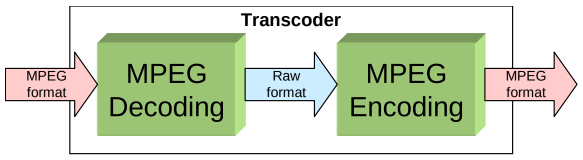

Fig. 5: Transcoder Phases

- *decode* MPEG stream into a raw format (YUV, YV12, PPM, etc)
- *encode* the raw format back into MPEG video, at the new encoding parameters, after an initial processing (adjust frame size, change frame rate, bitrate)

The encoding step demands most processing power (in general about 3-4 time more than decoding, for the same encoding) and is heavily dependent on the encoding parameters (e.g. frame size). [14]

If we denote by $\boldsymbol{L}$ the average CPU processing power (percentage load) for performing a task, then the requirement for transcoding from initial format $\boldsymbol{f_i}$ to final format $\boldsymbol{f_f}$is :

$\boldsymbol{L_{trans}(f_i, f_f) = L_{dec}(f_i) + L_{enc}(f_f)}$, where $\boldsymbol{fs, fr, br}$ specify frame size, frame rate and bit rate respectively of the MPEG encoding format and $\boldsymbol{L <= 1}$. $\boldsymbol{L_{dec}}$ and $\boldsymbol{L_{enc}}$ denote the proxy load for the decoding and encoding (intermediary steps) in real time.

The middleware continuously monitors the system and ensures that enough resources are available to serve all clients. For example, when $\boldsymbol{n}$ users are connected to a proxy, the total load on the proxy node should not exceed the available computing resources: $\sum_n \boldsymbol{L^i_{trans} <= 1}$, where $\boldsymbol{L^i_{trans}}$ represents the load required for user $\boldsymbol{i}$ (in the case of a cluster of $\boldsymbol{N_P}$ proxies, total load should be less than $\boldsymbol{N_P}$).

4.2 Network Bandwidth Limitation

As the wireless network bandwidth is limited, a large number of users performing simultaneous video streaming may lead to network congestion and the inability to provide the users with an acceptable quality video stream (above the user defined threshold).

Network traffic regulation was discussed before for a single user case and is revisited here in a multi-user context.

Each individual client requires a burst transmission of length $\boldsymbol{P_b}$ bits every $\boldsymbol{I}$ seconds. It follows that the bandwidth consumption per user is given by:

$\boldsymbol{BW^i = P^i_b / I^i}$ (in *bps*), for client $\boldsymbol{C^i}$.

If the total wireless bandwidth available at the access point AP is $\boldsymbol{BW_{total}}$, then the condition for avoiding network congestion is: $\sum_n \boldsymbol{BW^i <= BW_{total}}$.

Note that we are considering the average network bandwidth, local communication spikes should be handled by AP, as long as the local buffering allows.

4.3 Multi-User Energy-Aware Admission Control & Stream Transformations

The middleware on the proxy utilizes the feedback from the devices, to continuously monitor the overall state of the system. Initially, for each client device under its control, it performs an energy aware admission control to identify whether a request can be scheduled (the video can be streamed at the user requested quality level for the entire length of the video). Subsequently, it monitors the residual energy at the device and streams the highest quality video (performing realtime video transcoding) that meets the energy budget at the device and maintains an acceptable QoS.

As this is a heterogeneous multi-user environment, with large diversity among various client devices, an information record is maintained that lists the device capabilities (in terms of power consumption, hardware characteristics, power management, battery information, etc). Moreover, to assist the control process, a configuration/mode table is created for each client device $\boldsymbol{C_i}$ which, for each video quality $\boldsymbol{Q_k}$, lists information available from the client, network and proxy level:

- architecture: optimal knob configuration (cache parameters, DVS mode) and average CPU power consumption at the architecture level for this configuration
- network: ideal burst length and corresponding network level power consumption
- proxy: load required for converting from initial stream to quality $\boldsymbol{Q_i}$
- access point: bandwidth for streaming video at quality $\boldsymbol{Q_k}$

Part of the information is sent by the actual device when it registers and the rest is precomputed by the proxy node middleware level. This table predicts the effect of streaming video quality $\boldsymbol{Q_k}$ to client $\boldsymbol{C_i}$ for all the involved components (client architecture - CPU and network card configuration and power consumption, proxy node - load, access point - bandwidth).

The system level admission control algorithm is presented in Fig 6. When a new client request or a feedback from a client is received, the proxy first tries to determine if it can accommodate the existing users at a minimum acceptable quality level (as specified by the users themselves). If not, the client is denied the request. Otherwise, starting from the minimum quality level, the middleware gradually increases the quality levels for all clients as long as all the constraints related to proxy load, AP bandwidth and available residual power on the individual devices are satisfied. When an optimized operating point for the system is determined as a result of this process, the new configuration is updated on the control system and the clients are notified about the changes (where changes exist).

The order in which clients are chosen by the algorithm as candidates for increasing quality levels is important for providing an overall improved user experience for all users. One possible heuristic would favor clients with a lower current quality, so that the system tends toward a uniform quality level coverage. When more clients are at the same quality level, the heuristic would select the one which incurs less delta change in proxy load and bandwidth as a result of a increase in the video quality by one level. In this way, possibly other users could be further served and the number of users for which a higher quality level is provided is maximized.

```
WHILE (TRUE)
{
  IF ($R_i$ OR $F$ received)
  {
    FOR each client
    {
      Determine $k$, such that $Q_k=Qa$
      IF $((T - (T_{cur} - T_{start}))* P_k <= E_{res}$                (1)
        Compute $L_{trans}$, $BW_{AP}$ for video quality $Q_k$
      ELSE
        REJECT the request OR (Negociate video quality)
}
    IF ( $\sum L^i_{trans} <= 1$ && $\sum BW^i_{AP} <= BW_{total}$)     (2)
    {
      WHILE (at least one change in the client configurations)
      {
        FOR each client $C_i$ (use heuristic to select)
        {
          Increase quality by 1: $Q_k$ <- $Q_k + 1$
          IF (Inequalities (1) and updated (2) hold)
            CHANGE quality to $Q_k$ for client $C_i$
        }
      }
      UPDATE system with new client configurations
    } ELSE
      REJECT the request OR (Negociate video quality)
  }
}
```

Fig. 6: Multi-user Admission Control

5 Performance Evaluation

We adopted a multi-phase methodology to achieve our results. First, through extensive experimentation and profiling we identified eight determinate video quality levels for a handheld. This determined our dynamic video transcoding parameters. Next, we optimized the low-level architecture to perform optimally with the above video streams. We also identified the best network transmission characteristics for video streams encoded at the above quality levels. Moving to a multi-user environment, we performed experiments to estimate the CPU load for transcoding between different formats. Using the operating points for architecture and network transmission, combined with the results from transcoding we used our proxy admission control algorithm to guide streaming videos to the iPAQ, and evaluate the performance for the system.

Video Quality	Cache Size	Cache Associativity	Clock Frequency	Voltage	Original Energy	Optimized Energy	Savings
Q1	8	8	100	1	1.29	0.76	47.5%
Q2	8	8	100	1	1.09	0.64	47.8%
Q3	8	8	100	1	0.95	0.56	48.0%
Q4	32	2	66	0.9	0.54	0.26	57.6%
Q5	32	2	66	0.9	0.48	0.23	57.8%
Q6	32	2	33	0.9	0.42	0.20	58.0%
Q7	8	8	33	0.9	0.29	0.14	57.3%
Q8	8	8	33	0.9	0.24	0.11	57.5%

Table 1: Architectural Configurations for Ideal Energy and Performance Gains (*Action* Clip)

5.1 Experimental Setup

All our measurements for video quality measurements were made for a Compaq iPAQ 3650, with a 206Mhz Intel StrongArm processor, with 16MB ROM, 32MB SDRAM. The iPAQ used a Cisco 350 Series Aironet 11Mbps wireless PCMCIA network interface card for communication. The batteries were removed from the iPAQ during the experiment and we measured power drawn by the device during MPEG streaming.

The CPU architecture simulation was implemented using the Wattch/SimpleScalar [3] power simulator. We configured our simulated CPU to resemble a typical Intel XScale processor (widely used in today's mobile devices, mostly due to their excellent MIPS/Watt performance): ARM core, 400 MHz, 1.3V, 0.18um, 32k instruction cache, 32k data cache, single issue. The MPEG decoder from Berkeley MPEG Tools [15] was used. Video transcoding was done using the commercially available TMPGEnc transcoder [18]. As input video for the decoding, we used traces from various video clips from low action(e.g. "news") like content to high action (e.g. GTA) fast scene changing streams. For each of these clips, we extract a sequence to be simulated and we encode it at noticeable different levels of quality . Level 1 corresponds to the best possible quality: 320x240 frame size, 30fps framerate, 650kbps bitrate. From 1 to 8, each level differs from the previous one by at least one parameter (frame size, frame rate or bitrate). This way, we have a clear (observable) degradation in quality between each quality level.

The decoding program is then simulated through Wattch and statistics are extracted from the simulator output.

5.2 Experimental Results

In this section, we first present the performance of our architectural and middleware optimizations, obtained in prior work. Our integrated framework is based on top of these individual results. Later, we present the improvements in the overall utility of the system achieved in the system with the integrated approach, in a distributed environment.

5.2.1 Architectural Optimizations

By profiling short (10sec) video clips through our power simulator, for all combinations of cache parameters (size: 4Kb to 64Kb, associativity: 1 to 32) we collected the total energy consumption for the entire duration of each video clip (with quality from 1 to 8)

For all video quality levels, we were able to determine a cache configuration that minimizes energy consumption. Moreover, through cache reconfiguration, we obtained power savings in the range 10-15%, depending on the nature of the video.

We made the following observations:

- Cache associativity produces the largest shift in energy consumption, extreme values proving very inefficient(especially for direct mapped caches).
- The best cache configurations reflect the internal storage requirements for different frame sizes and organization of the decoding algorithm.

5.2.2 Optimized Knobs after Applying DVS

We combined dynamic voltage scaling technique with cache reconfiguration for an increased overall effect on power consumption. The combined approach gave us up to 60% in energy savings as compared with the initial architecture. We repeated the same procedure for all the video quality levels.

In most of the cases we were able to run the CPU at a significant lower voltage, mainly due to the initial high speed of the simulated XScale processor (400MHz) and the quality of the code (highly optimized).

The overall energy savings we obtained after both above techniques and the knob values for the optimized configuration are summarized in Table 1, for an *action* video clip. These numbers serve as input of our integrated framework.

5.2.3 Middleware Optimization for Video Bursts

For each video quality (1-8), we varied the video burst time, the network noise level and the network packet size. The wired and wireless ethernet bandwidths were set to 10Mbps and 8Mbps(effective B/W, 802.11b is capable of higher throughput) and γ was set to a 0.85. The transmission delay of the wireless access point was also fixed at 400μs per packet.

As expected the highest quality video had a very small burst time compared to the lowest quality video. The ideal burst times were ascertained such that none of the frames missed deadlines at the device.

Table 2 shows the ideal burst times and the corresponding power savings for the same video stream encoded at the eight quality levels. We can observe that the power saved at the NIC, is the least for the highest quality video and the most for the lowest quality video. Also, as expected the power gains diminish with noise. Clearly, very small burst times yield no gains. Interestingly, for every additional user in the system, a new optimal burst time exists. The numbers we gathered are used to drive our integrated approach.

Quality Level	Burst Length (N=1, secs)	Power Saved (N=1,Watts)	Burst Length (N=3, secs)	Power Saved (N=3,Watts)	Burst Length (N=5, secs)	Power Saved (N=5, Watts)
Q1	2.3	.925	2.0	0.89	1.8	0.87
Q2	3.5	1.0	3.05	0.98	2.76	0.96
Q3	4.6	1.04	4.05	1.02	3.68	1.0
Q4	4.85	1.05	4.2	1.03	3.85	1.02
Q5	6.8	1.08	6.25	1.07	5.75	1.06
Q6	14.5	1.12	12.5	1.11	11.5	1.11
Q7	17.5	1.13	15.0	1.12	13.5	1.11
Q8	17.0	1.12	15.4	1.12	14.0	1.11

Table 2: Optimal network video burst lengths(in secs) and corresponding power gains for different quality and noise levels for the Grand Theft Auto action video, assuming sufficient buffer available at client and network packet size of 500KB

5.2.4 Transcoding Results

We perform transcoding experiments with the TMPGEnc transcoder. The original action type movie clip was converted using different parameters (frame size, bitrate, frame rate). By measuring the time required to perform the transcoding on a Intel P4 2.4Ghz, 512Mb internal memory, we make the following observations:

- Frame size (followed by frame rate) has the largest influence in transcoding time and hence, proxy load. This is expected, as the amount of processing required increases with the size of the frame to be processed.
- The bitrate does not change transcoding time significantly. Bitrate values control the quantization step in MPEG encoding, which is not computationally intensive.
- The nature of the video stream (action vs news type) slightly affects the total transcoding time, as seen in Fig 7, where we measured the transcoding time as a function frame size (relative to original clip) and video type

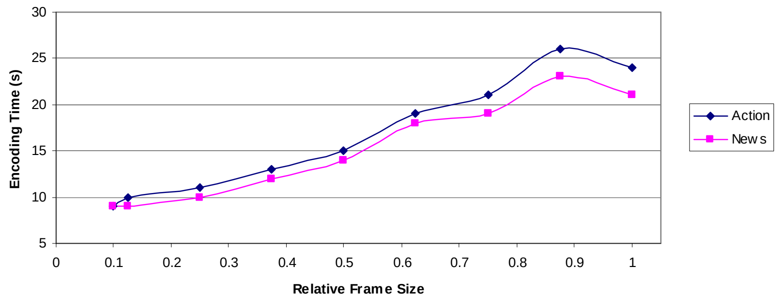

Fig. 7: Transcoding time for different frame sizes

Video	Proxy Load	
	Action	News
Q1	0.40	0.35
Q2	0.40	0.35
Q3	0.40	0.35
Q4	0.30	0.28
Q5	0.30	0.28
Q6	0.30	0.28
Q7	0.20	0.18
Q8	0.20	0.18

Fig. 8: Proxy loads for different videos

The transcoding time was used to compute the proxy load: if the total transcoding time when using 100% CPU utilization (single user mode) is t_{trans} (in seconds), the CPU load on the proxy system in a multi-user, real time transcoder is determined as: $\boldsymbol{L_{trans} = t_{trans}/60}$. Proxy loads for converting from the original format to our defined quality levels (Q1 through Q8) are presented in Fig 8 (for a 60 second video sequence).

5.2.5 The Integrated Framework

We finally evaluate the performance of the integrated framework with the architectural and middleware optimizations in place in Fig 9.

In order to generate a scenario that captures some of the problems in real environments we chose a set of constraints that make the system very sensitive to changes in the user configuration and performs a new adaptation on any important event (new user joins, power change feedback, MPEG stream finishes). Therefore we assume a system with one proxy node and a 1Mbps bandwidth available wireless network bandwidth. Each user specifies an acceptable video quality ($\boldsymbol{Q_a}$) of 7 or 8 (very low quality), to allow for a larger range of possible operating points.

Initially (T=0) there are 3 users in the system. Based on the available residual energy, the system decides on an initial video quality and device configuration for each user. At time $\boldsymbol{T_1}$ a new user request to join the system controlled by our proxy. In order to accommodate the new client, the middleware decides on lowering the video quality for the rest of the users in the network, as seen in the graphs. Note that all video qualities are still above or equal to the acceptable (threshold) video levels.

Due to unforseen circumstances user 1 is consuming energy at a higher rate than anticipated (for example, a new process was started on the actual device). As a result, at $\boldsymbol{T_2}$ the system lowers the quality to stream to client 1 and is able to increase the quality level for a different user (3 in our example).

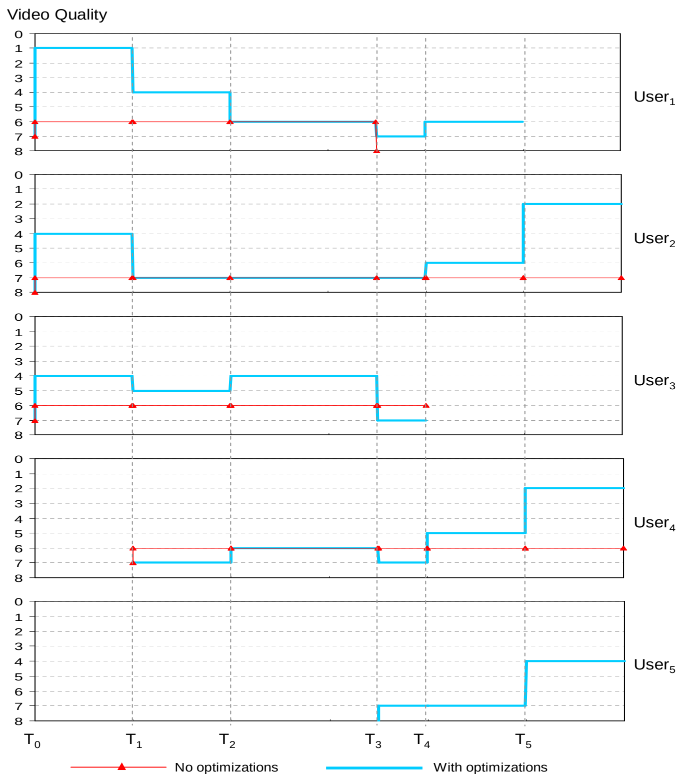

Fig. 9: Multi-user Scenario

At time $\boldsymbol{T_3}$, a fifth user joins the group. In order to accept the new user, the system is lowering the quality for all the user to decrease proxy node load and bandwidth usage. When user 3 finishes execution, the newly freed resources (proxy load, bandwidth) allow the system to improve on video quality for the other nodes.

Finally, when streaming finishes on client 1, more resources are released and the clients that are left can experience their video streaming at even higher levels.

As seen on the graph, without our optimizations and system adaptation, only 4 clients could be served simultaneously by the system. Moreover, due to the increased power consumption from other tasks, user 1 would not have been able to stream the entire video to completion. Even for the rest of the users, the possible video quality to be streamed was very low (between 6 and 8), while the dynamic adaptation could deliver quality above that most of the time. Note that there are situations where the system chose to lower the quality for some users to allow for a better overall global function.

Our integrated approach brings a significant improvement in the power consumption, which translates in a higher video quality that can be streamed and hence substantially improves the user experience at the client end. More users can be served and our dynamic adaptation is able to adjust the streams to meet the current state of the system.

6 Related Work

To provide acceptable video performance at the hardware level, efforts have concentrated on analyzing the behavior of the decoder software and devising either architectural enhancements or software improvements for the decoding algorithm. Until recently it was believed that caches can bring no potential benefit in the context of MPEG (video) decoding. In fact, due to the poor locality of the data stream, many MPEG implementations viewed video data as "un-cacheable" and completely disabled the internal caches during playback. However, Soderquist and Leeser [17] show that video data has sufficient locality that can be exploited to reduce cache-memory traffic by 50 percent or more through simple architectural changes. Dynamic Voltage Scaling [12, 6] for MPEG streams have been widely researched. At the application and middleware levels, the primary focus has been to optimize network interface power consumption [8, 4, 5]. A thorough analysis of power consumption of wireless network interfaces has been presented in [8]. In [16], Shenoy suggests performing power friendly proxy based video transformations to reduce video quality in real-time for energy savings. They also suggest an intelligent network streaming strategy for saving power on the network interface. The GRACE project [21] professes the use of cross-layer adaptations for maximizing system utility. They suggest both coarse grained and fine grained tuning of parameters for optimal gains. In [20], a resource aware admission control and adaptation is suggested for multimedia applications for optimal CPU gains. Dynamic transcoding techniques have been studied in [2] and objective video quality assessment has been studied in [19, 11].

7 Conclusions & Future Work

In this paper, we integrated low-level hardware optimizations with high level middleware adaptations for enhancing the user experience when streaming video onto handheld computers in a distributed environment. First we identified and fine tuned low level hardware to best perform with video streams at discrete quality levels. We then used a higher level middleware approach to intercept and transform the stream to compliment the architectural optimizations. A proxy based adaptive network transmission mechanism was developed to minimize the power consumption of the network interface card. The impact of handling multiple users on system resources was studied next. Finally, all the above techniques were integrated into a system, and the overall system functionality was tested for a realistic scenario. Significant improvements were observed in the requested video stream quality, enhancing the user experience substantially. We are currently exploring further architectural, middleware and system level adaptations for improving the power consumption of displays, storage devices etc. and integrating them into the framework.

References

[1] "ITU-R Recommendation BT-500.7, Methodology for the subjective assessment of the quality of television pictures". In *ITU Geneva Switzerland*, 1995.

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

[3] David Brooks, Vivek Tiwari, and Margaret Martonosi. Wattch: A framework for architectural-level power analysis and optimizations. In *ISCA*, June 2000.

[4] Surendar Chandra. Wireless Network Interface Energy Consumption Implications of Popular Streaming Formats. In *MMCN*, January 2002.

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

[6] Kihwan Choi, Karthik Dantu, Wei-Chung Chen, and Massoud Pedram. Frame-Based Dynamic Voltage and Frequency Scaling for a MPEG Decoder. In *ICCAD 2000*, 2002.

[7] Radu Cornea, Shivajit Mohapatra, Nikil Dutt, Alex Nicolau, and Nalini Venkatasubramanian. Integrated power management for video streaming to mobile handheld devices. Technical Report 03-19, University of California, Irvine, 2003.

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

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

[10] Chung-Hsing Hsu, Ulrich Kremer, and Michael Hsiao. Compiler-directed dynamic frequency and voltage scheduling. *Lecture Notes in Computer Science*, 2008:65–??, 2001.

[11] Jan Jansen, Toon Coppens, and Danny De Vleeschauwer. Quality Assessment of Video Streaming in the Broadband Era. In *ACIVS*, 2002.

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

[13] Shivajit Mohapatra and Nalini Venkatasubramanian. PARM: Power-Aware Reconfigurable Middleware. In *ICDCS-23*, 2003.

[14] Darsan Patel and Wansik Oh. Performance profile of MPEG-2 transcoding with motion vector reuse mechanism. Technical report, Kent State University, 2001.

[15] Ketan Patel, Brian C. Smith, and Lawrence A. Rowe. Performance of a software mpeg video decoder. In *ACM Multimedia*, 1993.

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

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

[18] TMPGEnc. http://www.tmpgenc.net.

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

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

[21] W. Yuan, K. Nahrstedt, S. Adve, Doug. Jones, and Robin Kravets. Design and Evaluation of a Cross-Layer Adaptation Framework for Mobile Multimedia Systems. In *MMCN*, January 2003.