

# Managing Cross-Layer Constraints for Interactive Mobile Multimedia\*

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 in a real environment, under power, bandwidth and load constraints is a significant design challenge, due to the various capabilities of these devices and the real-time character of the streaming workloads. A unified framework that integrates low level architectural optimizations, OS power-saving mechanisms and adaptive middleware techniques 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 managing such a distributed, real-time system, while meeting the constraints imposed by the environment and maintaining acceptable QoS levels for each client. We demonstrate how the framework, which 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. The heavy computation and real-time nature of multimedia workloads place a heavy burden on these already energy-constrained devices. In addition, human perception of multimedia quality is significantly influenced by the device specific attributes (e.g form factor) [13]. Therefore, delivering high quality multimedia content to mobile handheld devices, while preserving their service lifetimes remain competing design requirements. Moreover, distributed environments where heterogeneous devices perform simultaneous streaming introduce new problems related to shared resources, and QoS trade-offs for accommodating an increased number of users.

Over the years, different solutions have been proposed at various computation levels that try to optimize for power and performance and still meet the system constraints. However, there has been less effort towards integrating these techniques with a final goal of managing the overall system under multiple constraints at each of these levels.

Fig. 1 presents the different computation levels in a typical handheld computer and the cross layer interactions for optimal power and performance deliverance. The hard-

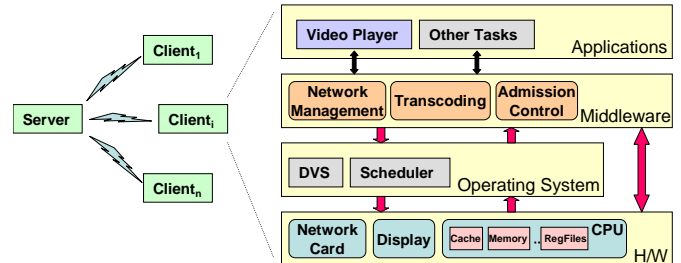


Fig. 1: Abstraction layers in a distributed multimedia streaming application

ware level includes the physical components of the handheld: CPU, memory, display, network card. The OS level handles the task scheduling and dynamic voltage scaling of the CPU. The middleware coordinates the system in a distributed environment and the application level includes all the media players and other programs running on the device.

The three main components that draw power in the handheld device are: the *CPU*, the *network* and the *display*. Therefore, we aggregate hardware and software techniques that induce power savings for these resources.

We have adopted a multi-phase approach. First, low-level architectural tuning knobs are identified and optimized for video streams of predetermined quality levels; we then 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 (under power, load and network bandwidth constraints). Finally, experiments with the integrated approach are presented, with 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 meet the system constraints while still providing an improved user experience.

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

## 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.

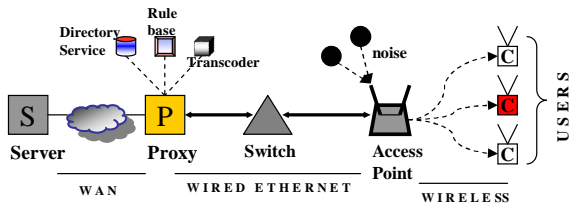


Fig. 2: System Model

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). 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.

In previous work [13] 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, 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 real-time 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, followed by a fine grain adaptation of the architecture to the video stream.

## 3 On-device Optimizations

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].

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: display (around 1W for full backlight), network hardware (1.4W) and CPU/memory (1-3W, with the additional board circuits). We started by focusing our attention on the possible optimizations at the CPU level for a multimedia streaming application (MPEG).

Simulation shows that internal CPU caches account for the largest part of the energy consumed by the processor. There are various optimization techniques applicable to cache. 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).

Our cache reconfiguration goal is optimizing energy consumption for a particular video quality level  $Q_k$ . 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.

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. 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 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  $D$  for a particular video stream/quality. The difference between the average frame delay and actual frame decoding time gave us the slack time  $\theta = F_d - D$ . 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  $\theta$ .

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 tech-

nique (cache configurations with a larger 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.

## 4 Global Optimizations

The savings obtained from the lower levels (architecture) can be further amplified if combined with techniques on higher levels (middleware). We present here our existing work on intelligent network transmission. 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.

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.

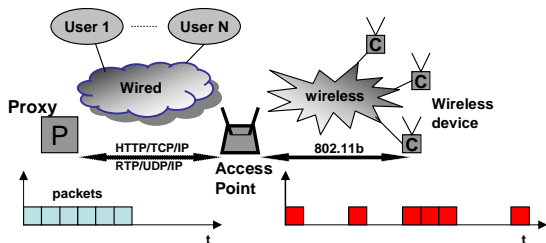


Fig. 3: Wireless Network

The analysis for the above power saving approach was performed using a realistic network framework (Fig. 3), 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 queuing 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 external noise). The above analysis could be used by an adaptive middleware to calculate an optimal burst length for any given video stream and noise level.

## 5 System-level Optimizations

So far we have studied multimedia streaming/adaptation in a scenario where a single client was served by the proxy

node. Extending our system architecture to multiple users unfolds new problems, like network congestion, proxy node resource exhaustion, etc.

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:

- *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]

The middleware continuously monitors the system and ensures that enough resources are available to serve all clients. For example, when  $n$  users are connected to a proxy, the total load on the proxy node should not exceed the available computing resources:  $\sum_n L_{trans}^i \leq 1$ , where  $L_{trans}^i$  represents the load required for user  $i$  (in the case of a cluster of  $N_P$  proxies, total load should be less than  $N_P$ ). This is a simple way of ensuring that the proxy will not be overloaded. More accurate scheduling in the presence of multi-proxy nodes is subject of future work.

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).

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

$$BW^i = P_b^i / I^i \text{ (in bps), for client } C^i.$$

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

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

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.

The system level admission control algorithm is presented in Fig 4. 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

```

WHILE (TRUE)
{
  IF ( $R_i$  OR  $F$  received)
  {
    FOR each client
    {
      Determine  $k$ , such that  $Q_k=Q\alpha$ 
      IF  $((T - T_{cur} - T_{start}) * P_k \leq E_{res}$  (1)
      Compute  $L_{trans}^i, BW_{AP}$  for video quality  $Q_k$ 
      ELSE
      REJECT the request OR (Negotiate video quality)
    }
  }
  IF  $(\sum L_{trans}^i \leq 1 \ \&\& \ \sum BW_{AP} \leq BW_{total})$  (2)
  {
    WHILE (at least one change in the client configurations)
    {
      FOR each client  $C_i$  (use heuristic to select)
      {
        Increase quality by 1:  $Q_k \leftarrow Q_k + 1$ 
        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 (Negotiate video quality)
}
}

```

Fig. 4: Multi-user Admission Control

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.

## 6 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. 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.

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 CPU/architecture simulation for determining optimized cache configurations 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 for decoding the MPEG stream. For modeling the real-time constraints, we verified that the simulation time for the decoding was within the required deadline. We computed the best DVS voltage/frequency based on the actual simulation time and the required deadlines from the video stream. We assume that changes in quality occurs only at scene boundaries and are not very frequent, so the overhead of architecture reconfiguration is very small.

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.

Because we propose techniques and architecture enhancements which are not presently implemented in handhelds, we combine results from simulation (for the CPU architecture and network card) with real measurements of the energy drawn by the iPAQ handheld.

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.

### 6.1 Local Optimizations

By profiling short (10sec) video clips through our power simulator, 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 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.

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. Note that these are savings over the CPU and memory, as given by the power simulator.

### 6.2 Global Optimizations

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  $\gamma$  was set to a 0.85. The transmission delay of the wireless access point was also fixed at 400 $\mu$ s per packet.

As expected the highest quality video had a very small burst time compared to the lowest quality video. The ideal

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*)

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. 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.

We perform transcoding experiments with the TMPGENC transcoder. The original action type movie clip was converted using different parameters (frame size, bitrate, frame rate). We measured the time required to perform the transcoding on a Intel P4 2.4Ghz, 512Mb internal memory

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:  $L_{trans} = t_{trans}/60$ .

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

**Table 3:** Proxy loads for different videos

We finally evaluate the performance of the integrated framework with the architectural and middleware optimizations in place in Fig 5.

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 ( $Q_a$ ) of 7 or 8 (very low quality), to allow for a larger range of possible operating points.

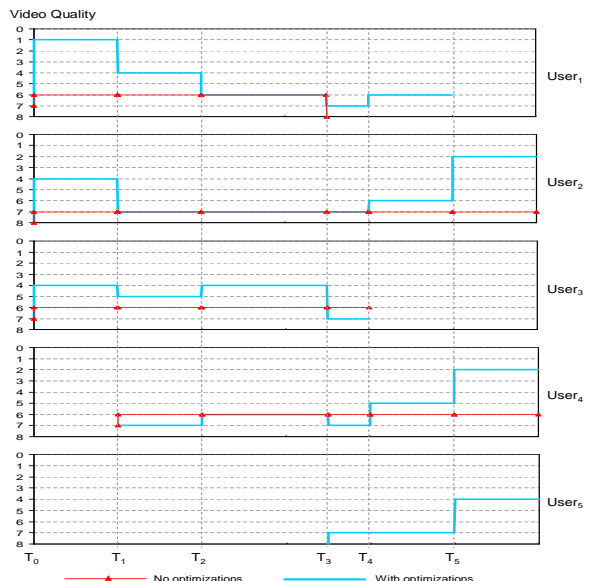
Initially ( $T=0$ ) there are 3 users in the system. Based on the available residual energy, the system decides on an initial video quality and device configuration for each user. At time  $T_1$  a new user request to join the system controlled by our proxy.

Due to unforeseen 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  $T_2$  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).

At time  $T_3$ , 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.

Our integrated approach brings a significant improvement in the power consumption. More users can be served and our dynamic adaptation is able to adjust the streams to meet the current state of the system.



**Fig. 5:** Multi-user scenario

## 7 Related Work

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

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

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.

The GRACE project [21, 20] professes the use of cross-layer adaptations for maximizing system utility. They suggest both coarse grained and fine grained tuning through global co-ordination and local adaptation of hardware, OS and application layers. The coarse/global adaptations are expensive and less frequent and occur only when global system changes are triggered (e.g task-set changes). The local adaptations are for the local variation in the execution of tasks. In GRACE, the global and local coordinators exist on the local device and perform the necessary adaptations. In our work, we use a proxy based distributed middleware approach, that integrates cross-layer (architecture, OS, middleware, application) adaptations on the local device with distributed adaptations such as adaptive traffic shaping and transcoding at the proxy for energy gains.

Dynamic transcoding techniques have been studied in [2] and objective video quality assessment has been studied in [19, 11].

## 8 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. 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. Watch: 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, Radu Cornea, Nikil Dutt, Alex Nicolau, and Nalini Venkatasubramanian. Integrated power management for video streaming to mobile handheld devices. In *ACMMM*, November 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.