Dynamic Stream Control for Energy Efficient Video Streaming

Martin Kennedy, Hrishikesh Venkataraman and Gabriel-Miro Muntean
Performance Engineering Laboratory
Dublin City University (DCU)
Dublin, Ireland
martin.kennedy@ieee.org, hrishikesh@eeng.dcu.ie, munteang@eeng.dcu.ie

Abstract— **Gartner predicts that by 2013 mobile devices will overtake PCs as the most popular type of device used for accessing Internet-based services. Meanwhile, mobile phones are becoming increasingly complex and powerful. However, battery technology has not been increasing at the same pace and hence, there is an urgent need for solutions to balance the battery performance of mobile devices and their functionality. An efficient solution would be to analyze the application(s) running on the device and manipulate all available device parameters, in order to maximize the power saving while minimizing the effect on the Quality of Service. As an initial step towards this unified solution, this paper takes an extremely important functionality as a starting point - *video streaming*. Different mechanisms for adaptive multimedia streams in mobile devices are investigated. Furthermore, an enhanced version of the Battery and Stream-Aware Adaptive Multimedia Delivery algorithm (BaSe-AMy) is tested on a real-world mobile device. The experiments result in an increase to the maximum playtime of a video stream of up to 10%, as compared to non-energy-aware streaming solutions, while also maintaining a high stream PSNR value.**

Keywords-mobile, portable and handheld devices; display technology; video coding and processing; VoD, interactivity, datacasting;

I. INTRODUCTION

Both the functionality and the computation capability of mobile devices have increased exponentially in recent times. Importantly, modern smart-phones and PDAs are now capable of interacting with multiple sensors, multitasking and communicating over various independent network interfaces. While these devices follow a functionality improvement rate similar to Moore's law, developments in battery life have lagged behind considerably. A classic example of the gap between functionality and power-supply is the iPhone 4. When used continuously, for web browsing over 3G, the battery life lasts a mere 6 hours [1].

Mobile video streaming is an area which is experiencing an incredible growth rate. According to *Allot*, the first quarter of 2010 saw a 92% increase in video streaming applications. These applications account for 35% of the total global mobile bandwidth usage [2]. The specifications of the HTC Nexus One promote a battery life of 7 hours during talk time in 3G networks [3] but our measurements have shown that for the energy intensive application of video streaming, the battery depletes in as little as 4 hours. In order for a mobile phone to be effectively used on a day-to-day basis, it has to be able to function for at least 12 hours before requiring battery recharging. Currently, there are different energy conservation techniques included in mobile Operating Systems (OS), such as background process control and ambient light-aware screen brightness adaptation. However, these approaches are not sufficient to provide the required energy savings in the device.

An intelligent, context-aware algorithm is a prerequisite for dynamically adapting the characteristics of a device in an energy and application-aware manner. Each type of application in a device would have different Quality of Service (QoS) requirements and may also involve different methods for user interaction. As a result, each application type will require different methods for achieving reductions in the power consumption. Each application type needs to be analyzed independently and subsequently incorporated into a total solution. To this end, this paper investigates energy-optimizations that can be achieved on a mobile device, in the context of an adaptive video streaming application. In particular the paper analyzes the benefit brought by the Battery and Stream-Aware Adaptive Multimedia Delivery algorithm (BaSe-AMy) in terms of energy savings in comparison with a non energy-aware video streaming solution.

The remainder of this paper is organized as follows. In Section II different energy saving mechanisms in video streaming applications are discussed. Additionally three available adaptive streaming solutions for mobile devices are surveyed. Following that, the architecture of the system and an enhanced version of the BaSe-AMy algorithm are introduced in Section III [4]. The test set-up and results are presented in Sections IV and V and then conclusions and future work are explained in Section VI.

II. RELATED WORKS

Lately, video streaming applications have been studied in detail. Recently, it has been shown that for video streaming applications on wireless mobile devices, the screen, the processor and the network interface offer the largest range for possible energy savings ([5], [6], [7], [8]). This is illustrated in Figure 1. A number of different approaches can be taken in order to minimize the energy consumption of a mobile device during video streaming. Some of these are discussed below.

Adaptive Video Decoding

Adaptive decoding refers to operations on the client device that alter the default decoding process in real-time in order to

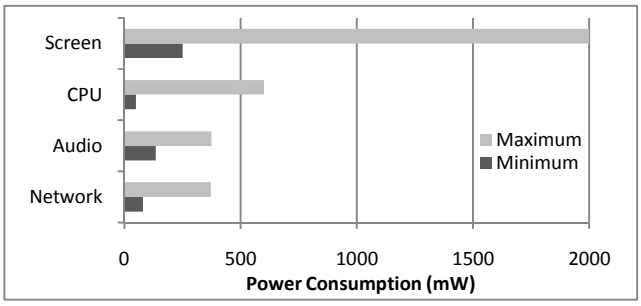

Figure 1 – Variation in Power Consumption of Components on Nexus One Android Phone

maximize energy efficiency. This can involve simplifying the decoding process or skipping specific video frames instead of decoding them. This process inherently lowers the QoS but also increases battery life.

By changing the quantization parameter for each macro-block in the video decoder on the client device, Park *et al.* achieved a 42% decrease in energy consumption during video-decoding with a mere 13% quality degradation in the video [9]. In [10], Yu *et al.* propose an algorithm for scaling the frame-rate of a video sequence during the video decoding process for reductions in decoding time and power consumption. The algorithm assesses the level of movement between the immediately preceding decoded video frames. If the level of movement between the previous frames is above a certain threshold, then the current frame is decoded as normal. However, if the level of movement is below the threshold, then the current frame can be discarded without decoding it and its reference frame is displayed again. This yielded a reduction in decoding time by up to 35.9% while the PSNR of the video dropped from 37.44 dB to 33.12 dB

It should be noted that each of the aforementioned mechanisms for adaptive video decoding is complementary to the work presented in this paper. The BaSe-AMy algorithm specifically looks at adaptive video encoding and delivery

Display Control & Dynamic Voltage Scaling (DVS)

Different display technologies have very different energy consumption characteristics. Cheng *et al.* proposed an algorithm for LCD screen devices to adapt the backlight of the screen while also compensating for the degradation in video quality by adjusting the intensity level of each pixel in the video frame [11]. The algorithm yielded power savings of over 40% while still maintaining a "fair" QoS. This approach requires additional processing at a video proxy but ideally it could be realized as a terminal-based solution on the device while still yielding some energy reductions.

In contrast, OLED screens do not utilize a backlight for the illumination of the display. The power consumption of OLED screens depends on the intensity level and chromaticity of each pixel [12]. Dong *et al.* were the first to begin research on manipulating the colors of pixels on OLED screens in order to conserve energy on mobile devices [13]. The result of this work was the application "Chameleon" which dynamically altered color in the context of a web-browsing application. The power consumption for displaying each color on the OLED screen of a device is modeled in the application. This model is then used for altering the color scheme of websites to schemes that have a higher energy efficiency rating. Initial tests showed a power reduction of 41% on the device when using the "Chameleon" application.

In [14], Yang *et al.* proposed an algorithm for dynamically scaling the voltage supply to a mobile device's CPU. The decoding time for each video frame is predicted and used to select a frequency level on the CPU that will successfully decode a certain ratio of frames in time for presentation on the screen. This algorithm resulted in a reduction of system-wide energy consumption by up to 17% over other DVS mechanisms and up to 24% over non-DVS mechanisms.

The dynamic display control and dynamic voltage scaling techniques are complementary to the work in this paper. However, a middleware implementation is necessary to achieve each technique. This paper solely looks at optimizations that can be implemented in the application layer.

Adaptive Video Streaming Mechanisms

There are currently three standards that have been adopted for implementing adaptive streaming of MPEG-4 H.264 videos to mobile devices.

1. Apple has developed HTTP Live Streaming (HLS) [15]. HLS is an open standard which Apple has submitted to the IETF [16]. HLS works by taking a video input and encoding it at multiple different levels of bit-rate. Each of these levels of video are then segmented into multiple sections of uniform playback duration. A mobile client can request a section of video over a HTTP connection. Subsequent sections can be downloaded as required for playback. At any point the client can switch over to request video sections from one of the other levels of bit-rate. The result is that an adaptive stream is delivered to the mobile device. HLS is currently available on iOS devices since version 3.0. One advantage of HLS is that it is not tied to any single platform for either delivery or consumption. Adobe recently added HLS to Flash Media Server (FMS) [17], so that FMS can also be used to delivery video streams to devices that are not Flash-enabled. Google has also added HLS to Android 3.0 [18] which makes it a ubiquitous solution.
2. As well as supporting HLS, Adobe has their own dynamic streaming solutions: RTMP Dynamic Streaming and HTTP Dynamic Streaming. RTMP Dynamic Streaming does not require any segmentation of the video streams but it does require the use of FMS for the stream delivery. HTTP Dynamic Streaming [19] can be served from FMS or an Apache server. This approach works in a similar way to HLS in that it requires segmentation of the video stream before transmission. Both these solutions would be played in a Flash player or an Adobe AIR application. While flash players are available on most platforms, requiring their utilization is quite a limiting restriction. Any development of streaming applications would have to be done using proprietary Adobe software.

3. Notably, Microsoft has developed a system called Smooth Streaming [20] which is part of their Silverlight system. Smooth Streaming uses HTTP as the delivery protocol for its streams. Although this mechanism is not compatible with Android phones, Microsoft have released solutions for both iOS and Windows Mobile. One important thing to note is that developers do not have complete access to the configuration of the adaptation algorithms [21].

Section IV details the selection of the adaptive streaming solution used in the work reported in this paper.

III. ARCHITECTURE

The BaSe-AMy algorithm that is proposed in this paper is an evolved version from the first incarnation, as seen in [4]. The algorithm assesses the remaining battery level, the remaining video stream duration and the packet loss rate and then dynamically adapts the stream bit-rate and display brightness to maximize energy conservation without sacrificing the Quality of Experience (QoE) too much. The current version of the algorithm has been updated to adapt the display brightness dynamically. The algorithm also assesses the loss rate more accurately and the threshold for the loss rate has been altered for a value that is *more applicable in a real-world scenario*. The value for the loss threshold in this paper is modeled on Adobe's Dynamic Streaming Class [22] that triggers automatic adaptation if the packet loss rate goes over 25%. The different levels of thresholds for different battery levels in the original algorithm have been abandoned in the enhanced BaSe-AMy algorithm. The reason for this is the thresholds jeopardize achieving the maximum playback duration which can result in user dissatisfaction with the service. In the assessment for adapting up again, a new mechanism has been introduced to smooth out the video adaptations and reduce the ping-pong effect. This is achieved by comparing the remaining battery life to an up-scaled level of the remaining stream duration.

This application has both client and server components as shown in Figure 2. The client is a wireless mobile device. When the user opens the video streaming application, the application cycles through the following stages:

1. Streaming

   The client immediately opens the video stream connection to the video server. The video stream is then displayed on the screen of the client device in full-screen mode. In the initial stages, the highest video quality stream is played.

2. Monitoring

   As the stream is playing, the remaining battery capacity, packet loss and stream duration are sampled periodically. The exact sampling period can be stipulated by the user or can be arbitrarily set in the algorithm before use on the device. Previous and current readings of the battery level are used to dynamically predict the remaining battery life.

3. Decision

   The enhanced BaSe-AMy algorithm (Figure 3) is implemented in the decision module on the client device. It analyzes the collected data from the monitoring stage and decides whether or not any adaptation is required. BaSe-AMy decides whether to adapt the video stream up or down one quality level or to adapt the device's display brightness in order to achieve optimal results.

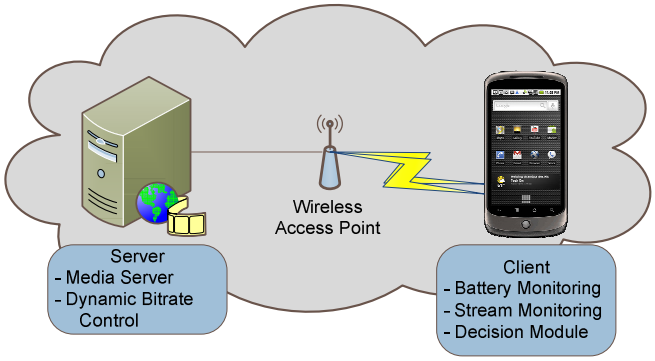

Figure 2 – BaSe-AMy Architecture

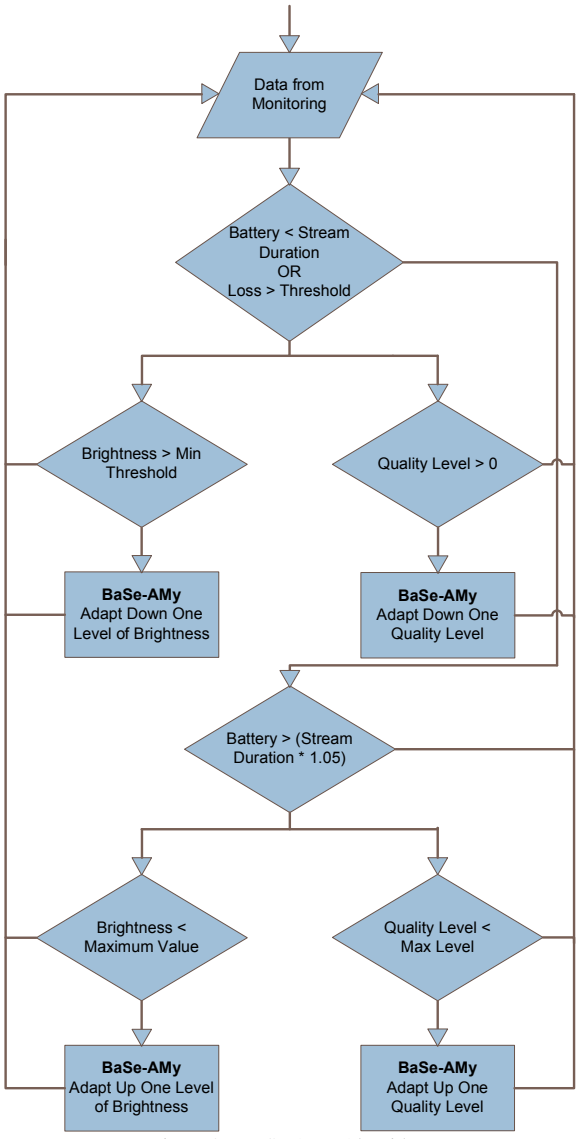

Figure 3 – BaSe-AMy Algorithm

Figure 4 – Test Setup: Server (laptop), Access Point and two Client Devices

4. Adaptation

When adaptations to the stream bit-rate are required, an order is sent to the server which seamlessly adapts the quality of the stream. Alterations to the level of brightness of the display are implemented in the video streaming application on the client device. The application then resumes monitoring its resources again.

IV. TESTING AND SETUP

The complete testing setup can be seen in Figure 4. This mechanism is evaluated using a HTC Nexus One running Android 2.3 as the client device in an 802.11g network. This specific device was selected because of its wide range of functionality and because it runs Android, an Open Source platform. Additionally, Android has integrated functionality for logging the power consumption of the device, which is ideal for the use-case, considered in this paper.

Adobe's RTMP Dynamic Streaming mechanism is selected for the implementation of adaptive video streaming to the client. This has been selected because it is a viable option across a wide range of mobile and laptop Operating Systems. Apple's HLS would be the ideal choice of mechanism but unfortunately the adoption of this technology by other companies has just begun. While there are reports of a HLS implementation on Android 2.3 from Nextreaming [23], neither the player nor the SDK are currently available online. Therefore, Adobe's Flash Media Server 4 is used to serve the adaptive H.264 video stream to an application written in Adobe AIR and compiled for Android. The power battery percentage of the Nexus One is measured periodically by the application and these values are used to predict the remaining battery-life. The remaining duration of the stream and the loss can be obtained from the stream meta-data in Flash. Adaptations to the brightness of the device's display cannot be implemented in an Adobe AIR application as this functionality is not exposed in the API. In order to calculate the total energy savings on the device, this issue was circumvented by running the Adobe AIR application and measuring the energy consumption for each

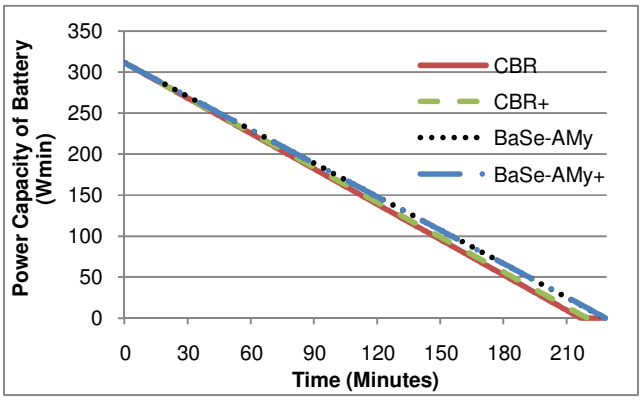

Figure 5 – Test 1 - Battery Capacity over Time

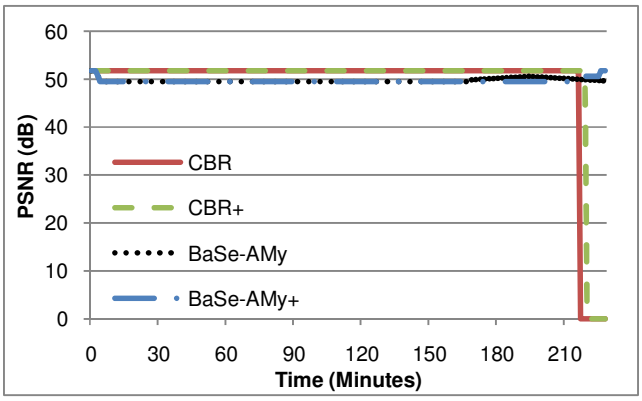

Figure 6 – Test 1 - PSNR over Time

level of video quality and brightness combination. This information was logged and later compiled for analysis.

Two separate tests were performed to assess the value of the BaSe-AMy algorithm. The video that was streamed during these tests was a 113s high-action video clip that was played on loop for the desired stream duration. This clip was encoded at 15 frames per second, with a resolution of 800x480 pixels using the H.264 codec and an MP4 container. The video was trans-coded to 5 different levels of bit-rate: 2 Mbps, 1525 kbps, 1.05 kbps, 575 kbps and 100 kbps. It is important to note that Adobe AIR on Android has poor performance for video stream playback with videos encoded above a bit-rate of 500 kbps. As Apple HLS becomes available on Android, there will be no need to use the Adobe mechanism as in this paper. However these tests are performed here to illustrate the potential for energy savings using the BaSe-AMy algorithm.

Four streaming mechanisms are compared in the tests presented here. The first involves a Constant Bit-Rate (CBR) 2 Mbps stream delivered to a device whose screen brightness is set to 100%. The second makes use of the same CBR stream, but the device has the default OS automatic brightness control enabled. The third mechanism is the BaSe-AMy algorithm with the device screen brightness statically set to 60%. The final mechanism is the BaSe-AMy algorithm with dynamic screen brightness control. Each mechanism is compared in terms of remaining battery capacity and QoS.

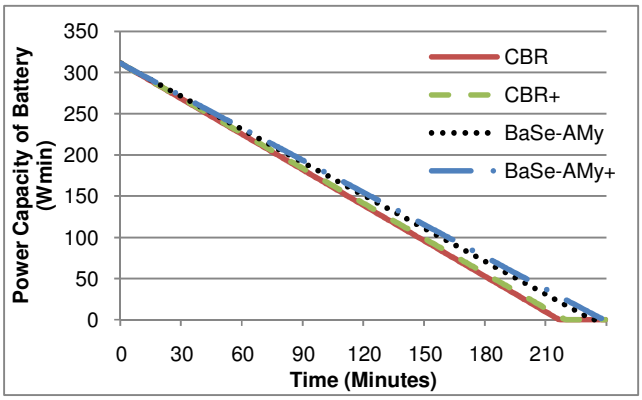

Figure 7 – Test 2 - Battery Capacity over Time

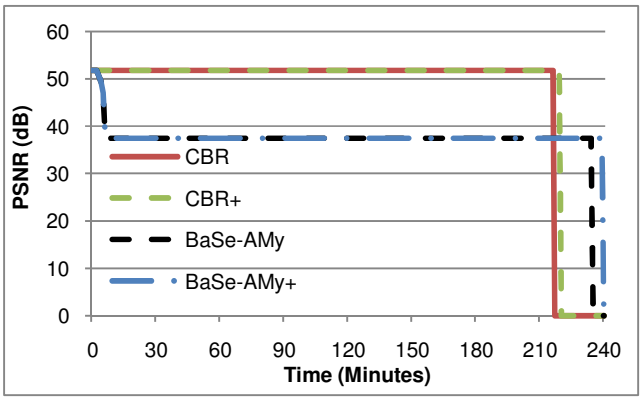

Figure 8 – Test 2 – PSNR over Time

The first test involves streaming 228 minutes of video from the media server to the server to the client device. In the second test a 235 minute video is streamed.

V. Results & Analysis

The results of each of the test scenarios are plotted in Figures 5-8. The labels CBR+ and BaSe-AMy+ are to signify when screen brightness control is enabled. For CBR+, this is the default system scheme but for BaSe-AMy+, the brightness is adaptively controlled in the BaSe-AMy algorithm.

1. In the first test, the CBR and CBR+ mechanisms result in total battery depletion before the whole video sequence can be displayed. However both the BaSe-AMy and BaSe-AMy+ mechanisms enable the full video stream to be played completely due to the adaptivity.
2. In the second test, the duration of the video stream was increased. While the CBR and CBR+ mechanisms resulted in total battery depletion in the same length of time as seen in the first test, the BaSe-AMy and BaSe-AMy+ algorithms successfully adapted the stream playback to increase the battery life by 7.6% and 10.1% respectively.

Table I provides a comparison of results from each of the two tests.

TABLE I. Comparison of Streaming Mechanisms

					Gain
Test 1 Video Length: 228 Mins	Battery Life	CBR vs CBR+	12978s	13159s	1.39%
		CBR vs BaSe-AMy	12978s	13680s	5.41%
		CBR vs BaSe-AMy+	12978s	13686s	5.45%
	PSNR	CBR vs CBR+	49.05dB	49.73dB	0.68dB
		CBR vs BaSe-AMy	49.05dB	49.66dB	0.61dB
		CBR vs BaSe-AMy+	49.05dB	59.57dB	0.52dB
Test 2 Video Length: 235 Mins	Battery Life	CBR vs CBR+	12978s	13159s	1.39%
		CBR vs BaSe-AMy	12978s	13965s	7.6%
		CBR vs BaSe-AMy+	12978s	14289s	10.1%
	PSNR	CBR vs CBR+	46.8dB	47.45dB	0.65dB
		CBR vs BaSe-AMy	46.8dB	37dB	-9.8dB
		CBR vs BaSe-AMy+	46.8dB	37.7dB	-9.1dB

VI. Conclusion and Future Work

This paper focuses on assessing the benefits of an improved version of the recently proposed BaSe-AMy algorithm tested on an Android-based device. The implementation investigated in this paper is achieved using Adobe RTMP Dynamic Streaming to an Adobe AIR application that was compiled for Android. While stream playback in the Adobe AIR application is poor for high-quality videos, experimental results show the potential for increasing the battery life by up to 10% while watching a video stream.

The future work will focus on implementing an adaptive streaming solution on mobile devices using Apple HLS. Additionally, further development of the unified energy-aware algorithm is required. This would include aggregating the BaSe-AMy algorithm with other energy saving techniques. The development of this unified algorithm would also involve researching methods for energy conservation for other application types and would require the construction of a combined intelligent, context-aware algorithm.

Acknowledgment

The authors would like to acknowledge Enterprise Ireland (EI), Science Foundation Ireland (SFI) and Dublin City University for their support.

References

[1] Apple Inc., "Apple (Republic of Ireland) - iPhone - Technical specifications of iPhone 4." [Online]. Available: http://www.apple.com/ie/iphone/specs.html. [Accessed: 08-Apr-2011].

[2] Allot Communications Ltd., *Allot MobileTrends Global Mobile Broadband Traffc Report, H2, 2010*. 2011.

[3] "HTC - Products - Nexus One - Specification." [Online]. Available: http://www.htc.com/www/product/nexusone/specification.html. [Accessed: 04-May-2011].

[4] M. Kennedy, H. Venkataraman and G.-M. Muntean, "Battery and Stream-Aware Adaptive Multimedia Delivery for Wireless Devices," in *IEEE International Workshop on Performance and Management of Wireless and Mobile Networks (P2MNET)*, Denver, USA, 2010.

[5] D. Theurer, M. Kennedy, H. Venkataraman, and G.-M. Muntean, "Analysis of Individual Energy Consuming Components in a Wireless Handheld Device," *China-Ireland International Conference on Information and Communications Technologies (CIICT)*, Wuhan, China, Oct. 2010.

[6] A. N. Moldovan, A. Molnar , C. H. Muntean, "EcoLearn: BatteryPower Friendly e-Learning Environment for Mobile Device Users", *Learning-Oriented Technologies, Devices and Networks, A. Lazakidou and I.Omary (Eds),Lambert Academic* Publishing, pp.273-296, 2011

[7] C.H. Muntean, “Improving Learner Quality of Experience by ContentAdaptation based on Network Conditions‘, *Computers in Human Behavior Journal,Special issue on ’Integration of Human Factors in Networked Computing”*, Vol.24, No. 4, pp. 1452-1472, 2008.

[8] A.N. Moldovan, C.H. Muntean, “Personalization of the MultimediaContent Delivered to Mobile Device Users,” *Broadband Multimedia Systems andBroadcasting, IEEE International Symposium on*, Bilbao, Spain, 2009

[9] S. Park, Y. Lee, J. Lee, and H. Shin, “Quality-adaptive requantization for low-energy MPEG-4 video decoding in mobile devices,” *Consumer Electronics, IEEE Transactions on*, vol. 51, no. 3, p. 999–1005, 2005.

[10] W. Yu, X. Jin, and S. Goto, “Temporal scalable decoding process with frame rate conversion method for surveillance video,” *Advances in Multimedia Information Processing-PCM 2010*, p. 297–308, 2011.

[11] L. Cheng, S. Bossi, S. Mohapatra, M. E. Zarki, N. Venkatasubramanian, and N. Dutt, “Quality adapted backlight scaling (QABS) for video streaming to mobile handheld devices,” *Networking-ICN 2005*, pp. 662–671, 2005.

[12] M. Dong, Y. S. K. Choi, and L. Zhong, “Power modeling of graphical user interfaces on OLED displays,” in *Design Automation Conference, 2009. DAC’09. 46th ACM/IEEE*, 2009, p. 652–657.

[13] M. Dong and L. Zhong, “Chameleon: A Color-Adaptive Web Browser for Mobile OLED Displays,” *Arxiv preprint arXiv:1101.1240*, 2010.

[14] A. Yang and M. Song, “Aggressive dynamic voltage scaling for energy-aware video playback based on decoding time estimation,” in *Proceedings of the seventh ACM international conference on Embedded software*, Grenoble, France, 2009, pp. 1-10.

[15] Apple Inc., “HTTP Live Streaming Resources – Apple Developer.” [Online]. Available: http://developer.apple.com/resources/http-streaming/. [Accessed: 09-May-2011].

[16] Apple Inc., “draft-pantos-http-live-streaming-06 - HTTP Live Streaming.” [Online]. Available: http://tools.ietf.org/html/draft-pantos-http-live-streaming-06. [Accessed: 09-May-2011].

[17] Kevin Towes, “Sneak Peek: Future Adobe technology for HTTP streaming across multiple devices Â« Kevin Towes on Online Video at Adobe.” [Online]. Available: http://blogs.adobe.com/ktowes/2011/04/sneak-peak-future-adobe-technology-for-http-streaming-across-multiple-devices.html. [Accessed: 09-May-2011].

[18] “Android 3.0 Platform | Android Developers.” [Online]. Available: http://developer.android.com/sdk/android-3.0.html. [Accessed: 09-May-2011].

[19] Adobe Systems Incorporated, “HTTP Dynamic Streamin Datasheet.” [Online]. Available: http://www.adobe.com/products/httpdynamicstreaming/pdfs/httpdynamicstreaming_datasheet.pdf. [Accessed: 09-May-2011].

[20] A. Zambelli, “IIS smooth streaming technical overview,” *Microsoft Corporation*, 2009.

[21] David Hassoun, “Dynamic streaming in Flash Media Server 3.5 – Part 1: Overview of the new capabilities | Adobe Developer Connection.” [Online]. Available: http://www.adobe.com/devnet/flashmediaserver/articles/dynstream_advanced_pt1.html. [Accessed: 09-May-2011].

[22] Adobe Systems Incorporated, “media server software tools and downloads | Adobe Flash Media Server.” [Online]. Available: http://www.adobe.com/products/flashmediaserver/tool_downloads/. [Accessed: 13-May-2011].

[23] Nextreaming Corp., “NEXTREAMING - Mobile Multimedia Leader.” [Online]. Available: http://www.nextreaming.com/product/nexplayer.php. [Accessed: 14-May-2011].