

# SMERT: Energy-Efficient Design of a Multimedia Messaging System for Mobile Devices

Lin Zhong\*  
Rice University  
Houston, TX 77005  
lzhong@rice.edu

Bin Wei\*  
AT&T Labs-Research  
Florham Park, NJ  
bw@research.att.com

Michael J. Sinclair  
Microsoft Research  
Redmond, WA 98052  
sinclair@microsoft.com

## ABSTRACT

Customized multimedia content delivery has become one of the most desirable applications to mobile device users. However its intense usage of wireless and user interfaces poses a great challenge to device usability and battery lifetime. In this paper, we describe our design and implementation of an energy-efficient multimedia messaging system to address this challenge. We construct a hierarchical system for users to access multimedia content, leveraging widely available short message service (SMS), an embedded system-based new interfacing device, and the Internet capability of mobile devices. Being the first of its type, the new system not only reduces energy overhead but also improves the usability of the service.

## Categories and Subject Descriptors

C.2.1 Network Architecture and Design; H.5.1 Multimedia Information Systems; H.5.2 User Interfaces

## General Terms

Measurement, Design, Human Factors

## Keywords

Energy-efficient design, mobile services, multimedia messaging

## 1. INTRODUCTION

Delivering multimedia messages [1] to a mobile device extensively uses its wireless and user interfaces, which are known to be two of the most power-hungry components [2, 3]. Our measurement and analysis showed that 90 multimedia messages could easily exhaust the energy supply of a commercial Smartphone.

In this paper, we present the design and implementation of an energy-efficient multimedia messaging system, called SMERT, to address the above problems. Our contribution is to utilize a hierarchical system for message delivery and user interfaces. SMERT is hierarchical in two aspects. First, the message content is hierarchical. A message can be delivered as text, or enhanced with key-frame images or video clips with different resolution and frame rates. In

our system, the messaging server generates a message in different formats but sends to the mobile user only a short text message. The text message includes links to other feature-rich formats. Upon receiving the message, the mobile device automatically determines which format to fetch. Second, we use a wrist-worn low-power display device for the user to retrieve text messages to minimize the usage of the power-hungry display on the mobile device. After receiving a text message, the mobile device can determine whether to notify the user immediately or send a message to the wrist-worn display. All these are done in a battery-aware and prioritized fashion. To the best of our knowledge, our hierarchical multimedia messaging system is the first of its type.

It is worth noting that while most other low-power system design work focuses on the device itself, e.g. better power management or dynamic voltage scaling policies, our work extends to the whole ecosystem in which a mobile device operates. It involves servers, middleware, embedded systems, mobile devices, and user interfaces. As we discuss in the paper, such an extension reveals great energy-saving opportunities.

In the rest of the paper, we first provide background information in Section 2, and describe our energy-efficient approach and the SMERT system in Section 3. We present the core techniques behind SMERT in Section 4. We summarize the design lessons learned from this work for multimedia mobile services in Section 5, discuss related work in Section 6, and conclude in Section 7.

## 2. BACKGROUND

Before presenting our techniques, we use this section to provide background information about the mobile device with which we conduct this study and describe an existing multimedia messaging system, MediaAlert [1], on which this work is built.

### 2.1 Power Profile of a GPRS Smartphone

Although cellular phones can enjoy the 3G data services and WiFi, GPRS is the most widely available service at the time of this writing. In this study, we use an Audiovox SMT5600 Smartphone with GPRS from Cingular, as a typical mobile device. Table 1 presents its system information as is pertinent to our study.

Table 1. System information of Audiovox SMT 5600

Wireless	GPRS Class 10 and Bluetooth
OS	Windows Mobile 2003 SE
Display	2.2 inch, 176 x 220 TFT LCD with 64K Colors
Battery	1064mA-Hour
Audio	Integrated loud-speaker

\*With equal contributions

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

DAC 2006, July 24–28, 2006, San Francisco, California, USA.  
Copyright 2006 ACM 1-59593-381-6/06/0007...\$5.00.

We obtained power status by measuring the voltage drop across a 100mΩ sense resistor embedded in the power supply cord with a USB-1608FS module from *Measurement Computing* [4]. Table 2 summarizes the data we collected. Note that “Idle” power is the basic case when the system is idle with display off. Other cases are extra power consumption required by the corresponding components to the idle case. For example, the entry of “System busy” is the extra power consumption when the system is repeatedly carrying out discrete-cosine transforms (DCTs) as compared with the idle case. Note that loudspeaker power consumption can vary significantly, depending on the duty cycle incurred by the sound. The LCD lighting power data is for typical luminance for nighttime reading. The GPRS power also can vary significantly, depending on the network condition. Bluetooth TRANSFER is the power used by Bluetooth transmission at 115Kbps data rate.

Table 2 demonstrates how power-hungry the user interface (display and loudspeaker) and wireless communication are. Such a profile is typical for most Smartphones and handheld devices.

## 2.2 MediaAlert Multimedia Messaging System

MediaAlert[1] is an industrial prototype service that delivers customized multimedia content to mobile users. It automatically monitors a large number of TV content feeds, extracts the content that matches user-interest profiles, repurposes the content, and delivers repurposed content to users according to their device profiles. The system consists of a media processing platform and a content delivery platform.

**Table 2. Power profile of Audiovox SMT 5600**

Component		Extra power (mW)
Idle		20
System busy		370
LCD		13
LCD Lighting		Between 56-212
GPRS		Up to about 1600
Bluetooth	PENDING	Between 1-3
	TRANSFER	Up to about 300
Loudspeaker		About 45

The *media processing platform* continuously records selected broadcast TV programs from several broadcasters using satellite or cable feeds based on a pre-determined schedule and according to the interests of the target audience. The structured video feeds from broadcast television are then digitized, compressed and stored in a database. The record is then processed to identify the content that is relevant to each user. A content repurposing module processes the content and generates a presentation that fits the target device. An *interest profile* consists of a number of topics, each of which is defined by a set of keywords. The delivery method as well as the content presentation depends on the user’s *device profile*, which specifies the device capacity such as display and connectivity. The user provides both profiles via a Web interface. The *content delivery platform* has gateways that manage the interfaces to mobile devices with a multitude of protocols and servers. The mobile devices can range from a simple numeric pager to handheld devices capable of playing video.

For example, when the MediaAlert system finds that a news story matches a user’s interest profile, it extracts the video clip, repurposes the content according to the user’s device profile, and delivers the message to the user through the standard multimedia messaging service (MMS), if the device is an Audiovox SMT5600. Upon receiving the message, the phone notifies its user, who then takes out the phone and downloads the entire MMS message. Such a process, unfortunately, is not only interruptive but also power-hungry as we will see next.

## 2.2 Energy Cost of Multimedia Messages

Table 3 shows the energy cost in the Audiovox Smartphone by a message (Video of 70 seconds) from MediaAlert on CNN’s coverage on eBay’s acquisition of Skype. It also shows those by the text and key-frame versions. Note that the exact energy cost may vary, depending on external factors such as cellular network quality and user behavior. The presented data were average over several measurements. “Downloading energy,” energy consumption by downloading the message, is different for automatically fetching and manual downloading. Manual downloading costs extra energy in display, which typically remains on when a user waits for the download to finish. “Consuming energy” is the energy consumption for the user to access the message, which is mostly consumed through display. We assume that text is consumed with a typical reading rate of 300 words per minute [5], four key-frame images are consumed with 4 seconds/frame, and video is consumed with loudspeaker on. Note that the MediaAlert system described in Section 2.2 provides entire messages in a predefined MMS format for cellular phones.

**Table 3. Energy cost by downloading and consuming different message formats**

Message Format		Text	Keyframes	Video
Size (Byte)		140	10K	696K
Downloading energy (Joule)	Automatic	~0.03	~1.2	~80
	Manual	~0.08	~1.7	~114
Consuming energy (Joule)		~5	~10	~48

Table 3 offers us many insights. First, the energy cost of a video message is extremely high. The Audiovox Smartphone will run out of battery if the user receives and consumes 90 such messages, even without any other usage. Second, the energy costs of different media formats differ drastically, by orders of magnitude. Therefore, it is possible to trade information richness for a longer battery lifetime. Third, manual downloading costs more than 40% more energy, mostly because of the need of the display. Fourth, “Consuming energy”, especially by display, dominates when the message is delivered as text or key-frames. These insights motivate us to improve the original MediaAlert system for better battery lifetime and usability.

## 3. SMERT SYSTEM

In this section, we describe the new system, called SMERT, which consists of several improvements over MediaAlert.

### 3.1 Hierarchical Message Content

For the MediaAlert service, a mobile end-user will have to download the whole MMS message to even get an idea of its content. Such an all-or-none policy incurs a huge energy waste. As

short text consumes much less energy, we can improve the MediaAlert system to provide multiple formats for the same message for the user to obtain information progressively. The system loads each format of the message onto a web server so that it can be retrieved with a unique URL. The system also generates a priority score for the message based on how well it matches the user interest profile. The system then delivers a text message to the user as an SMS, instead of an MMS message. Since SMS allows 140 bytes for a text message, we only include brief information about the content, the URLs for more detailed formats, message priority and lifetime.

### 3.2 Battery-Aware Message Fetching

Since the energy cost of receiving an SMS message is very low, a mobile device automatically retrieves it without interrupting the user. It then analyzes the message and uses a *battery-aware fetching policy* to determine whether the key-frames or the video of the message should be fetched before notifying the user. The policy is based on battery information, energy cost of fetching each format, and the message priority. It will be detailed in Section 4. A well-designed fetching policy will not only minimize the overhead due to manual downloading but also minimize interruption to the user.

### 3.3 Hierarchical User Notification

To address the interruptive and transient nature of MediaAlert message notification, we incorporated a low-power wrist-worn device, called CacheWatch [2, 6], as a secondary display to deliver text messages. Figure 1 shows the CacheWatch and its printed-circuit board (PCB) design. It can be associated with a host mobile device and serves it as a low-power remote display via Bluetooth. The current version is based on a TI MSP430 microcontroller, an OEM Bluetooth module, and a 96 by 128 dot-matrix LCD module from Epson. Without an operating system, it runs as interrupt-driven. It caches text messages from the host and displays them according to their meta-data. The CacheWatch can display a message notification in a non-interruptive, ambient fashion. Moreover, it takes user input through three series of touch sensors. The user can retrieve the text version of the message and instruct the mobile device to download a richer version of the message. Since the user does not directly operate the mobile device, the use of the power-hungry display is avoided.



Figure 1. CacheWatch and its PCB with a coin-size battery

There is, however, overhead associated with using the watch. First, the watch does not stay connected with its host as to minimize Bluetooth energy consumption. Every time they are connected, the host notifies the watch when to connect again. Therefore, when the host receives a message, it has to wait until the next connection to send the message notification to the watch, introducing a notification delay. Nevertheless, such a delay can be reduced and is often tolerable with multimedia content delivery. Second, the host mobile device has to pay energy cost in Blue-

tooth, which should be smaller than what we could save from using the watch instead of the host to retrieve text messages. Assuming it takes 20 seconds for a user to consume a text message on the Audiovox Smartphone, Figure 2 gives average phone power consumption when the phone synchronizes with the watch based on different intervals. It also shows that the numbers of 20-second phone accesses per hour that will incur the same average phone power consumption. For example, when the synchronization interval is 10 minutes, the watch only needs to reduce one text message access to the Smartphone every two hours to improve the latter's battery lifetime.

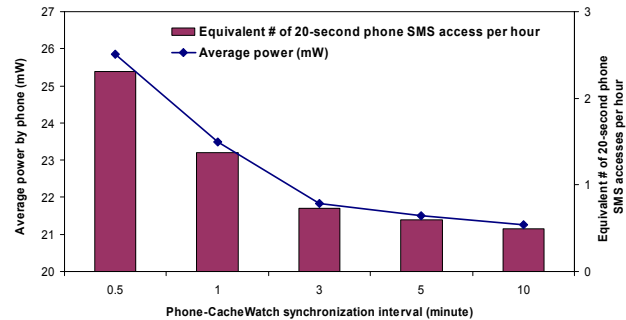


Figure 2. Power overhead from Bluetooth synchronization

We will address these two concerns again in Section 4.4. Before that, we present an overview of the new system next.

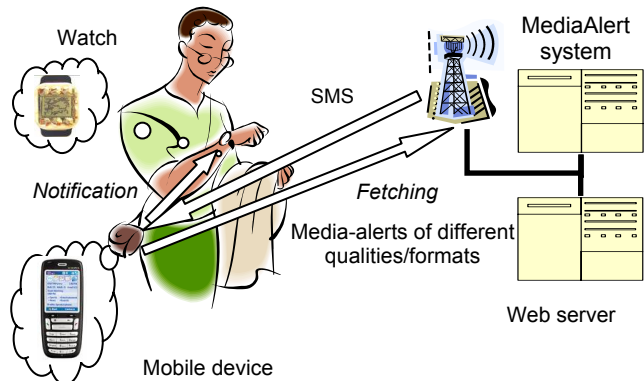


Figure 3. SMERT: A Hierarchical multimedia messaging system for mobile users

### 3.4 The New System

The new system, SMERT, implements the improvements described above. Figure 3 is an overall view of it. SMERT augments the original MediaAlert system with a web server for message retrieval and with the capability to generate multiple formats for a message. Most of the improvements can be implemented as software installed on mobile devices. In our prototype system, mobile users download and install software for progressive content delivery, battery-aware fetching and notification. They may choose to wear the watch, although the service works even without it.

We would like to emphasize that that SMERT provides more *mechanisms* than *policies* for mobile devices to save energy. It offers a lot of freedom for mobile device software developers to leverage such mechanisms as hierarchical content delivery, battery awareness, and wrist interaction, with their own policies.

## 4. CORE TECHNIQUES BEHIND SMERT

We next describe the core techniques behind energy-saving mechanisms of SMERT and related policy issues.

### 4.1. A Revisit of Energy Efficiency

Mobile devices have limited battery capacity but are recharged from time to time. Henceforth, the energy efficiency challenge is not to extend the battery lifetime as much as possible. Instead, it is to serve the user as much as possible with the available battery capacity before the next recharging. This implies the follows. First, a more desirable service should be allowed to consume more energy with higher priority. As a result, we need to prioritize multimedia messages. Second, remaining battery capacity at the time of recharging is simply a waste. It could have been used to better serve the user. Therefore, it is important to know the time of next recharging and the expected battery lifetime. These issues will be addressed in the following subsections.

### 4.2 Message Prioritization

To generate message priorities, we let users associate one of three urgency levels as an attribute with each keyword in SMERT. The three urgency levels are: critical, informative, and deferrable. The message priority score can be quantified by accumulating the contributions of all the matched keywords with level information and must satisfy the following properties:

- Between 0 and 1, and the larger, the more urgent.
- Matches of higher-urgency keywords result in higher priority.
- More keyword matches contribute to higher priority.

Since priority can be considered as the reverse of user's delay tolerance, we calculate a delay-tolerance score (DTS) as follows. First, each urgency level has an initial value, 1 for critical, 5 for informative, and 10 for deferrable, by default. Users can increase or decrease an initial value, indicating higher or lower delay tolerance for that level relative to other levels. Second, for the first match of a keyword in a level, it contributes the initial value of that level; the next match of the same level keyword contributes one less than the value of the previous match; when a keyword match makes the contribution value 0, all the remaining matches of this level will contribute 0. For example, for 3 matches of critical keywords, the contribution values are 1, 0, and 0; 3 matches of the informative are 5, 4, and 3; 3 matches of the deferrable are 10, 9 and 8. Assuming there are  $l$  matches of critical keywords,  $m$  of informative, and  $n$  of deferrable, we calculate the message DTS as:

$$DTS = \frac{\sum_{i=1}^l C_i + E \cdot \sum_{i=1}^m I_i + F \cdot \sum_{i=1}^n D_i}{d \cdot (l+m+n)}$$

where  $C_i$  is the contribution by the  $i$ th match of critical keywords;  $I_i$  is that by the  $i$ th match of informative;  $D_i$  is that by the  $i$ th match of deferrable;  $E=0$  if  $C_i > 0$  for some  $i$ , otherwise  $E=1$ ;  $F=0$  if  $C_i > 0$  or  $I_i > 0$  for some  $i$ , otherwise  $F=1$ ;  $d$  is the initial value of the deferrable keywords, or 10 by default. The priority score is finally calculated as  $(1-DTS)$ .

### 4.3 Battery Evaluation

In addition to message prioritization, battery information is also used in both fetching and notification policies. First, software on a mobile device reads the remaining battery capacity ( $RBC$ ) every 15 minutes and estimates the current energy consumption rate ( $ECR$ ) based on extrapolation. Second, it predicts the next battery charging time to derive an expected work time ( $EWT$ ), based on recharging

history or user specification. Then it calculates an energy-optimism score ( $EOS$ ) as

$$EOS = \frac{RBC}{ECR \cdot EWT}$$

The score is used in fetching and notification policies. Note that  $RBC/ECR$  is different from expected remaining battery lifetime, due to non-ideal battery discharging property. We only use  $EOS$  as an indication of how optimistically we should consume energy. We have also observed that  $ECR$  changes drastically along the day because mobile device usage changes too. Therefore, using the  $EOS$  to guide energy usage tends to smooth  $ECR$ . For example, when the user does not use the mobile device for a while and a low  $ECR$  is estimated, more energy consumption by multimedia messages will be allowed, which in turn increases  $ECR$ .

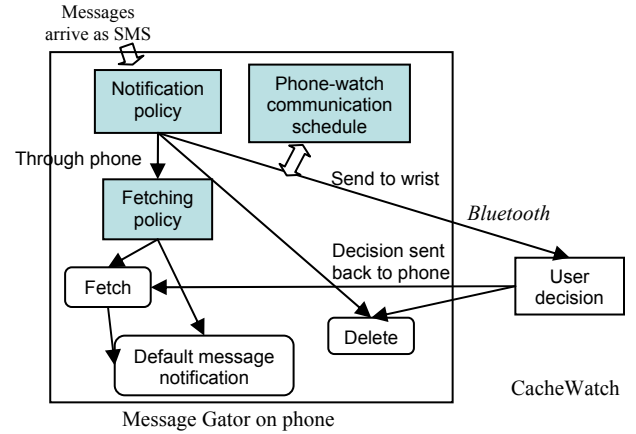


Figure 4. Message Gator

### 4.4 Battery-Aware Policies

Using the priority score and  $EOS$ , we devise preliminary policies to capitalize the energy-saving mechanisms described in Section 3. The policies are implemented as Smartphone software, called *Message Gator*. Figure 4 illustrates the message delivery process based on these policies. When a message arrives as an SMS, Message Gator automatically retrieves it. Message Gator first checks whether the message comes with a delivery instruction. If so, it simply follows it. Otherwise, it checks the *notification policy* to determine whether to notify the user through the watch or through the phone. In the latter case, Message Gator checks the *fetching policy* to determine whether to fetch the message with higher quality, to ignore the message, or to notify the user immediately, e.g. beep or vibrate. If the notification policy decides to notify the user through the watch, Message Gator waits for the next phone-watch communication to send a text message to the watch. Interacting with the watch, the user may confirm this message as read, or choose to fetch a higher-quality version. The decision will be conveyed to the phone upon next phone-watch communication.

#### 4.4.1 Battery-Aware Notification

Notification through the watch introduces delay in content delivery because the phone and the watch only communicate based on a schedule. As a result, our battery-aware notification policy notifies the user through the phone if and only if the priority score is greater than a threshold value,  $P_{TH}$ , which is set to be 0.5 in the prototype and can be changed by the user. The notification policy also serves as a battery-aware local filter to ignore low-priority messages when the energy-optimism score is low. The default thresholds for priority score and  $EOS$  are 0.1 and 0.2 in the prototype. They can be

changed by the user locally and complement the user interest profile in the MediaAlert system.

#### 4.4.2 Battery-Aware Fetching

A well-designed prioritization and battery-aware notification policy will route most of message notifications to the watch and let the user decide whether to download higher-quality versions after reading the text ones. Only a very few high-priority messages that require immediate user attention will be handled by the fetching policy, which examines

$$Y = P \cdot EOS - \frac{S}{10^6}$$

where  $P$  is the priority score,  $EOS$  the energy-optimism score, and  $S$  the size in byte of the smallest video format of the message (support for multiple video formats is not implemented yet for automatic fetching). If  $Y < 0$  and  $P \cdot EOS \leq 0.5$ , the user will be notified immediately without fetching; if  $Y < 0$  and  $0.5 < P \cdot EOS$  or  $0 \leq Y < 0.5$ , the user will be notified after the key-frames for the message are fetched; only when  $Y > 0.5$ , the user will be notified after the video clip for the message is fetched.

Since neither  $EOS$  nor  $P$  is greater than 1, the policy will not fetch a large video clip, especially when it is larger than 1MB. The policy implicitly makes a tradeoff between user attention consumption and energy consumption. That is, when the energy supply is not optimistic with a small  $EOS$ , the policy will use the user's discretion instead of automatically fetching higher-quality messages. The parameters in the policy are empirical determined and they can be adjusted by the user too.

#### 4.4.3 Adaptive Communication Schedule

As mentioned in Section 3.3, the more often does the mobile device communicate with the watch, the shorter it will take a message destined to the watch to show up on the wrist and the more energy will be consumed by Bluetooth. There are two ways to make a better tradeoff between the energy cost and delivery delay. First, if we have *a priori* knowledge of message traffic, we may use short communication intervals when traffic is high. For example, many TV programs are broadcasted based on a fixed schedule. If a user is interested in a certain program, he or she is more likely to receive messages in the corresponding program time, and can therefore set the mobile device to communicate with the watch more often during that period of time. Second, we can use a simple adaptive scheme to adjust the communication intervals based on observed message traffic. Currently, we use the following method. We first set the minimal and maximal communication intervals as 1 and MAX minutes, respectively, and start with an interval of 5 minutes. Suppose when the mobile device talks with the watch, there are  $N$  messages waiting to be sent and the previous interval was  $K$  minutes. Let  $K'$  denote the next communication interval in minutes. If  $N > 1$ ,  $K'$  will be set to the greater of 1 and  $K/N$  in minutes; if  $N = 1$ , it will be set to be the greater of 1 and  $(K-1)$ ; if  $N = 0$ , it will be set to be the smaller of MAX and  $(K+STEP)$ . STEP is a small number between 1 and 4.

We validated the effectiveness of the adaptive algorithm with synthetic message traces, which were generated with two assumptions: 1) messages arrive according to a *Poisson* process in a certain period of the day and 2) different time windows of the day have different density of message arrivals. Each trace covers multiple days. Each day consists of morning, afternoon, evening, and night, in which

messages arrive randomly according to *Poisson* processes with different densities. Figure 5 shows how the arrival time changes of a two-day trace. It also shows how the adaptive communication schedule technique successfully adapts to the trace with different parameters (MAX and STEP).

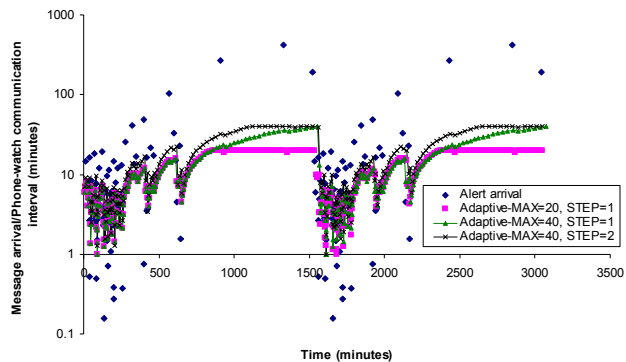


Figure 5. Message arrival and communication interval trace

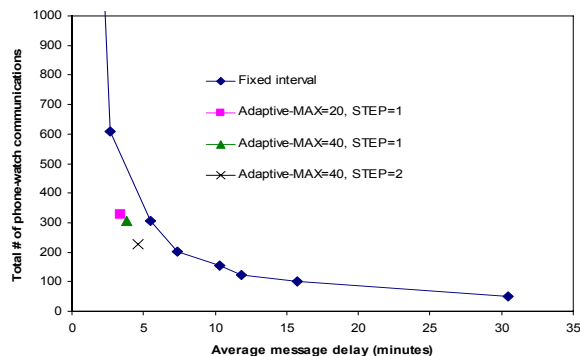


Figure 6. Tradeoffs between energy cost and message delay

Figure 6 compares the tradeoffs between energy cost, in terms of the number of device-watch communications, and the average message delay before being displayed by the watch. The solid line illustrates the tradeoffs made by using fixed communication intervals between 5 and 60 minutes. The tradeoffs made by our adaptive technique with different parameters are better than those by the fixed-interval approach in the *Pareto* sense. With the same average delay, the adaptive technique can save energy in device-watch communication by up to 40%. Although its effectiveness may vary with different message traces, we find that it almost always outperforms communication with fixed intervals. With about an average delay of 4 minutes, the mobile device and the watch communicate only about 300 times for two days, leading to about 1.3mW power overhead in the Audiovox Smartphone and about 2mW average power consumption by the watch. The 1.3mW Smartphone power overhead can be easily compensated, according to data presented in Figure 2.

## 5. DESIGN LESSONS

From SMERT, we have learned lessons that may apply to general multimedia mobile services, especially with regard to energy efficiency and usability.

**Looking beyond the mobile device itself offers great opportunities.** For most multimedia services, a mobile device is just the medium between the service/content provider, SMERT servers in this

study, and the service/content consumer, mostly the user. In SMERT, the hierarchical content delivery mechanism is primarily based on enhancement in the MediaAlert servers and the introduction of a CacheWatch. The mobile device basically implements policies to utilize this mechanism.

**Energy efficiency can be tightly coupled with usability.** Because display consumes significant power during human-computer interaction, energy efficiency can be improved by either improving user productivity or by replacing user engagement with automatic processes, e.g. the battery-aware fetching policy in SMERT. Moreover, since user behavior has a significant impact on system energy consumption, it is extremely important to conduct field investigation and user studies. We still need to further conduct user studies to see whether SMERT indeed delivers more services given the battery lifetime.

**Energy efficiency can mean different things in different application context.** Energy-efficient systems should serve their user best, given the battery lifetime. Therefore, it becomes extremely important to understand the user's need and the interaction between the user and the battery. For example, SMERT uses the battery recharging time to estimate the expected work-time for mobile devices. The battery-aware fetching policy can purposefully fetch more messages to increase the usability when remaining battery capacity is very high. It may consume more energy but deliver more services with the same battery capacity.

## 6. RELATED WORK

The services of monitoring media and alerting users are becoming very attractive on the Internet. CNN Alerts [7], Google Alerts [8], Microsoft .NET Alerts [9], and Yahoo Alerts [10] are monitoring news stories and deliver the selected contents to subscribers. Some provides the interesting areas; some allows users to choose their own keywords; some also support messaging to mobile devices. However, their delivery mechanisms are through established email accounts, web browser interface, Internet messenger, or text messaging to mobile devices. More and more service providers are adopting IP Multimedia Subsystems (IMS) as the preferred implementation for service delivery infrastructure. Messaging services can be quickly built to support a variety of mobile devices and protocols on the IMS-based networks, which tend to require more communication capabilities for mobile handheld sets. Thus how to increase the battery lifetime for communication is a severe problem. Our work provides the first systematic approach to address it.

Although SMERT is the first to utilize hierarchical content delivery, some past work shared a similar general philosophy. Flinn and Satyanarayanan investigated how to trade application data fidelity for energy savings [12]. While a SMERT mobile device outsources simple interactive tasks (text message retrieval) to a low-power wrist display, many researchers studied how computation could be offloaded from a mobile device to wall-powered computers [13-15]. Nevertheless, SMERT provides new energy-saving mechanisms for completely different mobile services.

## 7. CONCLUSIONS

Multimedia messages can easily exhaust the energy supply of a mobile device. To meet this challenge, we presented the design and implementation of an energy-efficient multimedia messaging service, SMERT. Instead of focusing on the mobile device, we examined the whole ecosystem in which mobile devices operate to reveal

more energy-saving opportunities. SMERT leverages the SMS and Internet capability of a mobile device to enable it to obtain the same content in different formats. Message Gator on the mobile device decides which format to fetch and how to notify its user based on information about the battery, message priority, and video clip size. Unlike that of most other work, our goal of energy efficiency is not to extend the battery lifetime as much as possible. Instead, our techniques seek to make the best use of battery capacity before recharging. Analysis and simulation has shown that our techniques are effective in achieving this goal. As extremely important to truly evaluate the energy efficiency and usability of the system, we are planning to conduct field investigation and user studies for SMERT. User studies may also shed insights into the parameter values that have been empirically set in SMERT so far. To the best of our knowledge, SMERT is the first reported hierarchical multimedia messaging system with extensive support for the energy efficiency of end-user mobile devices.

## ACKNOWLEDGEMENTS

This work was made possible in part through support from Enhancing Rice Through Information Technology program, a seed grant funded by Sheafor/Lindsay from the Computer and Information Technology Institute at Rice University.

## REFERENCES

- [1] Bin Wei *et al.*, "MediaAlert: a broadcast video monitoring and alerting system for mobile users," in *Proc. USENIX/ACM MobiSys*, June 2005.
- [2] Lin Zhong and Niraj K. Jha, "Energy efficiency of handheld computer interfaces: Limits, characterization, and practice," in *Proc. USENIX/ACM MobiSys*, June, 2005.
- [3] M. Stemm and R. H. Katz, "Measuring and reducing energy consumption of network interfaces in hand-held devices," *IEICE Trans. Communications*, E80-B(8), 1997.
- [4] Measurement Computing<sup>TM</sup>, <http://www.measurementcomputing.com>
- [5] R. P. Carver, *Reading Rate: A Review of Research and Theory*, Academic Press, Inc., San Diego, CA, 1990.
- [6] Lin Zhong and Mike Sinclair, and Niraj K. Jha, "A Personal-area network of low-power wireless interfacing devices: System & hardware design," In *Proc. ACM MobileHCI*, Sept. 2005.
- [7] CNN Alerts, <http://www.cnn.com/EMAIL/>.
- [8] Google Alerts, <http://www.google.com/alerts>.
- [9] Microsoft .NET alerts v6.0, <http://msdn.microsoft.com/msdnmag/issues/03/12/XMLFiles>
- [10] Yahoo Alerts, <http://help.yahoo.com/help/us/alerts/>.
- [11] IP Multimedia Subsystems (IMS), <http://www.3gpp.org/>.
- [12] Jason Flinn and M. Satyanarayanan, "Managing Battery Lifetime with Energy-Aware Adaptation," *ACM Trans. Computer Systems*, Vol. 22, No. 2, May 2004.
- [13] A. Rudenko *et al.*, "Saving portable computer battery power through remote process execution," *Mobile Computing and Communication Review*, 2(1), Jan. 1998.
- [14] Z. Li, C. Wang, and R. Xu, "Computation offloading to save energy on handheld devices: a partition scheme," In *Proc. ACM Int. Conf. CASES*, Nov. 2001.
- [15] Xiaohui Gu *et al.*, "Adaptive Offloading for Pervasive Computing," *IEEE Pervasive Computing Magazine*, July 2004.