# Energy-Aware Media Transcoding in Wireless Systems

Christian Poellabauer and Karsten Schwan
College of Computing
Georgia Institute of Technology
Atlanta, GA 30332
{chris, schwan}@cc.gatech.edu

## Abstract

*In distributed systems, transcoding techniques have been used to customize multimedia objects, utilizing trade-offs between the quality and sizes of these objects to provide differentiated services to clients. Our research uses transcoding techniques in wireless systems to customize video streams to the requirements of users, while minimizing the energy costs. We introduce an approach to dynamically determine which transcoders to execute and where to execute them (e.g., client or server). The goal is to select appropriate transcoders (a) to provide clients with the quality of service they desire while (b) minimizing the energy consumption of the end-hosts in accordance with application-specific global energy management directives. This paper investigates sample transcoder functions for video streaming on handheld devices and introduces a mechanism for selecting the most appropriate transcoders and transcoder parameters.*

## 1. Introduction

**Media Transcoding.** The proliferation of media streaming between resource-constrained wireless devices has raised the need for techniques that adapt these streams both to clients' quality of service (QoS) requirements and to the energy restrictions of battery-driven mobile systems. First, the heterogeneity found in the resources of mobile devices (e.g., different specifications for displays, processors, or network cards) requires that media streams are adapted to a device's capabilities. Second, dynamic variations in resource demands and in user requirements necessitate the run-time customization of distributed applications and of the services they utilize. In particular, with client-specific service differentiation, each client requires services to be adapted to its individual needs, thereby better matching the resources expended on service provision with client requirements and capabilities.

Transcoding is the process of transforming information from one form into another, e.g., to convert large images to smaller ones that are suitable for the limited resources of handheld devices or cellular phones. Frequently, the transcoding of data at one end-host of a client-server communication has consequences on the processing and communication requirements for both end-hosts. More than one transcoding function or set of transcoder parameters can be used to transform data into suitable forms, making it necessary to compare transcoders with respect to their potential provision of quality of service and energy savings.

**Energy-Aware Video Transcoding.** This paper introduces the concept of *global energy management directives*, which coupled with *energy-aware transcoding* provides both application-specific QoS and system-wide energy management. It further evaluates sample transcoder functions for video streaming applications and it introduces an approach to selecting transcoders and transcoder parameters. Global energy management directives are used to specify how *local* energy management techniques should be used to achieve and maintain some *global* energy goal. A sample global goal is that certain devices critical to a distributed application should be only minimally burdened with expensive (in terms of energy) tasks, whereas other, non-critical devices can be assigned larger burdens. In this paper, we address only application-level energy management techniques, i.e., applications adapt their behavior or the data streams they produce in order to modify their resource requirements and therefore their energy requirements.

An interesting aspect of the approach introduced in this paper is its ability to conserve energy by *adding* processing – in the form of transcoders – to a device. The intent is to reduce the energy consumption of a device by transforming large data elements into smaller ones, therefore reducing the costs of wireless data transmissions. However, such data transformations come at a price, i.e., the additional processing results in additional energy usage. As a result, data transformations reduce overall energy needs *only if* the ad-

ditional energy consumed for the execution of a transcoder is outweighed by energy savings due to the transmission of smaller data items and the reduced processing needs at the other end-host. Fortunately, for applications like video streaming, past work has resulted in the creation of many useful transcoders [14, 9], and there has also been substantial work on the selection of suitable transcoder parameters [23]. By leveraging such results, our work can focus on maintaining a client's desired QoS characteristics, using different transcoders that result in varying energy savings, depending on QoS specifications, device and transcoder characteristics, and data content.

Though applicable to many areas with large-data communications, wireless multimedia applications are an important target for energy-aware transcoding, for three reasons: (1) their increasing importance in mobile applications; (2) the fact that multimedia communications typically involve the long-running exchange of large data items, where data complexity or content change slowly over time; this justifies the potentially expensive deployment of transcoder functions; and (3) because these items can easily be changed in size and quality depending on resource availabilities and user needs. Video streams, as an example, can be customized in quality, including image size, color depth, resolution, or compression method. Here, different transcoders can be deployed, e.g., depending on a client's preferences, or transcoders can be made *parameterizable*, e.g., a 'resolution-transcoder' can be tuned with different parameters for the desired resolution. As a concrete example, consider two handhelds

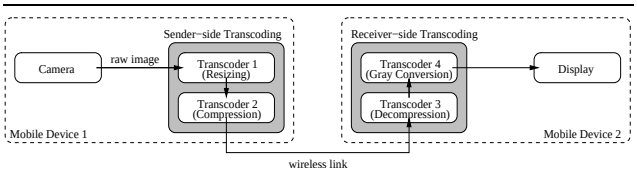


**Figure 1. Video streaming example.**

participating in a video communication, as shown in Figure 1. This could be part of an emergency communication system where firefighters and paramedics communicate in a disaster area using their handhelds, using visual communication to bring expertise wherever needed. Consider two devices used by medical personnel instructing each other in how to perform emergency care. One device captures raw images through its camera, which are then adjusted in size to fit the display of the receiver handheld and then compressed and transmitted to the receiver. At the receiver, these images are decompressed and further adjusted if required, e.g., converted to gray images if the device features a monochrome display only. In such cases, device battery life is critical and the decision about where transcoders are executed affects the energy requirements of both the sender and the receiver. Further, one can not assume the availability of nearby servers or support infrastructure. As a result, we assume that transcoders can only be placed at the end-points of a communication, i.e., intermediate points such as wireless access points and base stations are not available to users for the deployment of application-specific functionality such as video transcoders (e.g., in wireless ad hoc networks).

**Contributions and Related Work.** The main contributions of this paper are the concept of 'global energy management directives', coupled with the 'energy-aware transcoding' of continuous media streams in mobile systems. We evaluate sample transcoder functions for a multimedia application, and we introduce a transcoder selection and tuning approach that allows distributed systems to select the most appropriate transcoders and parameters, given clients' QoS requirements and the system's energy resources. Our approach combines off-line measurements of device and transcoder characteristics with the on-line prediction-based evaluation of desired QoS, current energy levels, and previous transcoder executions. Previous approaches have offloaded processing to other hosts, e.g., to extend the battery life on mobile devices [15] or to minimize energy costs in Internet data centers [18]. For mobile devices, researchers have developed energy management techniques, e.g., by addressing the energy costs of wireless data transmissions [19], by scaling device activity according to applications' resource needs [7, 21], or by switching between modes with different power characteristics [5]. The work introduced in this paper differs from previous work on quality-aware transcoding [3] in that our focus is on the reduction of energy consumption according to global energy management directives. Our work is similar to the one proposed by the authors in [4] in that they also address the use of transcoding to reduce energy consumption, however they focus on storage and energy limitations in an image capture device. Our work builds on similar work presented in [2], which investigates the trade-offs between processing costs of lossless compression algorithms and networking costs of transmitting reduced-size data. In contrast to that, this paper addresses more generally the transcoding of media streams in order to maintain application-specific QoS with particular focus on observing global energy management directives. Further, other approaches address the integration of multiple approaches to preserve energy [16, 12, 22], e.g., by coordinating adaptation across protection boundaries [25, 20, 8, 11]. The GRACE project [25, 24] proposes coordinated adaptation of hardware, operating system, and application layers to achieve fine-grained tuning

of system utility. In the Puppeteer project [10], a middleware framework is introduced that also uses transcoding to minimize energy requirements. The focus here is on closed-source applications, where the authors show that applications can significantly reduce energy usage by allowing applications and power management systems to incorporate knowledge of each other's activities. Our approach – in contrast to the related work discussed here – focuses on the application-level management of energy, by trading 'expensive' resources (in terms of energy) for 'cheap' resources. Further, feedback is used to improve the predictions made of future resource requirements. This is similar to the work presented in [17, 1], where the authors adapt network and CPU resources based on history, or in [13], where MPEG decoders are used to maximize a system's lifetime. Finally, in [16], the authors explore the use of transcoders for multimedia streaming, where transcoders reside on proxies. Our work is mostly concerned with fully mobile situations, i.e., both stream generation and replay are performed on battery-operated devices and devices can not rely on support infrastructure such as extensible and customizable base stations.

## 2. Global Energy Management

Previous work has pointed out the importance of *quality-aware* transcoding of multimedia [3] in order to provide differentiated services. In video transcoding, as an example, such transcoder functions can customize the video images to the restricted capabilities of mobile devices like handhelds or cellular phones. These transcoders can be classified into *mandatory* transcoders, i.e., transcoders that have to be executed by at least one end-host, and *optional* transcoders. Our goal is to use transcoders (a) to ensure that clients receive data in a form that corresponds to their QoS needs, such that (b) a chosen global energy management directive is observed. The following directives are supported:

**1. Maximize Sender's Operational Time (MAX-SOT).** The goal of this directive is to minimize the energy requirements of the *sender* of a media stream, i.e., all applicable transcoder functions are evaluated in order to find the ones that minimize the energy costs for the sender, without considering the consequences to the receiver. Figure 2 shows the scenario when the sender can preserve energy by exploiting transcoders. The first graph shows the energy costs for the transmission of the original data ($E_1$) and the transmission of some smaller-sized version of the same data ($E_2$). On the other hand, the second graph shows the processing costs for the execution of a transcoder to obtain the smaller-sized version of the data ($E_{cpu}$). If $E_1 - E_2 > E_{cpu}$, then the transcoder execution will result in reduced energy consumption at the sender.

**2. Maximize Receiver's Operational Time (MAX-ROT).** Here, the goal is to minimize the receiver's energy consumption, i.e., the sender will perform (a) all *mandatory* transcoders and (b) all *optional* transcoders that result in reduced energy requirements at the receiver.

**3. Maximize Application's Operational Time (MAX-AOT).** Here, the battery load levels of both sender and receiver have to be compared periodically and depending on their current levels, either MAX-ROT or MAX-SOT has to be followed. The goal of this directive is to keep the battery loads of the sender and the receiver at about the same level, in order to ensure that the distributed application can run as long as possible (i.e., both sender and receiver will run out of battery power at about the same time).

**4. Minimize System's Energy Consumption (MIN-SEC).** The energy requirements of the entire system are to be minimized, i.e., the combined energy consumption of sender and receiver has to be kept low. This is particularly important wherever energy consumption translates into costs (e.g., power supply for large hosting centers and cooling costs [6]). At the sender, this approach is similar to MAX-SOT, however, with one important difference: if the execution of a mandatory transcoder at the sender causes the sender to consume more energy compared to transmitting the original data, then the sender will still – unlike in MAX-SOT – execute the transcoder *if* the combined costs of the transmission of the original data and the costs for the execution of the transcoder at the receiver are greater than the processing and transmission costs at the sender.

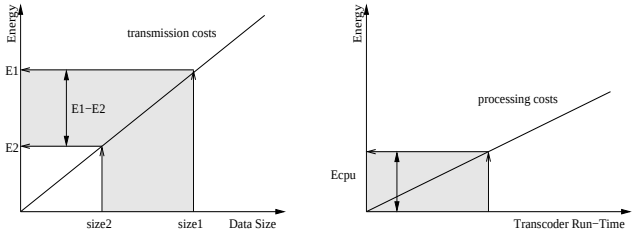


**Figure 2. MAX-SOT.**

Depending on the directive chosen, transcoder functions have to be evaluated and compared in order to determine the optimal transcoder function (or combinations thereof) for a given directive. This decision has to be re-evaluated frequently due to changes in application conditions, including changes in user requirements, resource allocations, or data content.

## 3. A Transcoding Framework

Our approach is based on the following observations. First, applications require that media streams have quali-

ties that are suitable for the limited capabilities of mobile devices, where these qualities can be expressed as 'QoS ranges' or as monotonically increasing utility functions. Second, transcoding is an appropriate technique in media streaming to adapt media quality to suit the needs of the clients, where lower quality media typically results in reduced communication needs. Frequently, multiple transcoding techniques can be applied, resulting in different qualities and data sizes. Third, in addition to the application- or client-specific needs for media quality, a system-wide goal in respect to energy needs has to be achieved. Figure 3

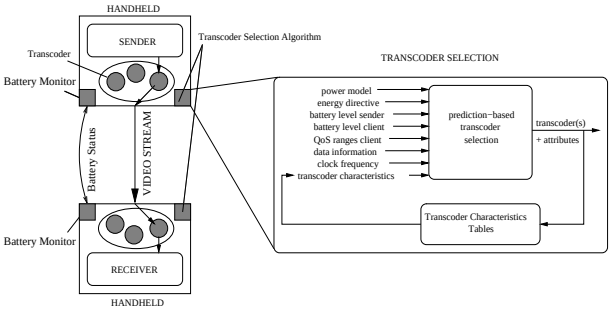


**Figure 3. Transcoder selection.**

shows the architecture of our approach. The sender transmits a media stream to a receiver, while the media stream is modified by a number of transcoders at either end-host. The transcoder selection algorithm collects a number of parameters in order to accurately predict the potential energy savings achievable by utilizing transcoder functions:

- **Clock frequency ($n$).** The clock frequency used at a device is required information since it affects the transcoder run-time. In all measurements in this paper, the clock frequency is kept constantly at the highest possible frequency.
- **Power model.** The device's power model is obtained through off-line measurements of the relationship between run-time and processing energy for each available clock frequency ($K_r(n)$) and the relationship between data size and transmission energy ($K_d(n)$).
- **Global energy management directive.** Sample directives are MAX-SOT, MAX-ROT, MAX-AOT, and MIN-SEC.
- **Battery load levels.** In the case of the MAX-AOT directive, the battery load levels of both sender and receiver have to be compared periodically (e.g., once per minute).
- **QoS specification.** The user specifies the desired quality of service, which is translated into parameters for the transcoders. These QoS parameters include the minimum image width and height, minimum color depth, or the frame rate. Note that the approach in this paper only addresses image-related qualities such as size or color depth; QoS parameters such as frame rate and jitter are managed separately by a rate-control approach, which is beyond the scope of this paper.
- **Transcoder characteristics.** Transcoder characteristics are determined off-line, i.e., the relationship between input data size and output data size (expressed as *ratio* $r_d$) and the relationship between input data size and transcoder run-time (expressed as *ratio* $r_r$), for each transcoder.
- **Data sizes $size(d)$ and $size(d')$.** The input data size ($size(d)$) is the size of some data $d$ to be transmitted before a transcoder function is applied, and the output data size ($size(d')$) is the predicted size of the output data (i.e., the size of the data after a transcoder is applied).
- **Transcoder run-time ($rt_t$).** This is the predicted run-time for each of the transcoders.

The transcoder selection is executed at both the sender and the receiver, while control information is attached to video frames (e.g., the most recent measured transcoder run-time) to ensure that accurate predictions are made. Further, the sender of a video stream indicates which transcoders have been executed for a given frame, such that the receiver can determine if further transcoder executions are required. In our approach, the transcoder selection algorithm is invoked for each frame, however, approaches are possible where the algorithm runs only at certain intervals, e.g., whenever the user changes the QoS requirements, or when the video frames change significantly in size or content.

**Obtaining Transcoder Characteristics.** In order to compare transcoders, the relationship between input data size and output data size ($r_d$) as well as the relationship between input data size and transcoder run-time ($r_r$) are required. In this paper, we measure these relationships off-line for sample input data and store it in tables. Each table entry contains the following information: *input data size*, *output data size*, and *transcoder run-time*. The transcoder selection framework predicts the output data size and the transcoder run-time by approximating these relationships based on the table entries. Different approximation methods exist, such as least squares regression. For simplicity, we approximate these ratios by searching the table for the input data sizes closest to a given data size (i.e., the nearest entry higher and the nearest entry lower) and then compute the mean value for the output data size and the run-time of these two entries. When a transcoder is executed, the actual transcoder run-times and output data sizes are determined and can be used to update table entries. To ensure that the entries for

transcoders that have not been used for a period of time are still accurate, unused transcoders can be executed periodically to obtain recent numbers for transcoder run-time and output data size, while considering these additional overheads in the transcoder selection process (this approach is not further discussed in this paper).

### 3.1. Computation of Potential Energy Savings

Next, we describe the approach of our framework to determine the energy savings that can be achieved by using the transcoders at the sender. A similar algorithm is used at the receiver, but is omitted for brevity. For each transcoder we obtain an estimated transcoder run-time from the input data size and the ratio $r_r$: $rt_t = r_r * size(d)$.

We then use the run-time - energy factor $K_r(n)$ to obtain the energy consumed by the execution of the transcoder at a particular clock frequency $n$: $E_t = K_r(n) * rt_t$.
Next, we obtain the output data size from the ratio $r_d$: $size(d') = r_d * size(d)$.
The output data size ($size(d')$) is used along with the factor $K_d(n)$ to determine the energy consumption for the transmission of the output data: $E_{d'} = K_d(n) * size(d')$.
Further, the input data size $size(d)$ is used to determine the energy consumption for the transmission of the original data: $E_d = K_d(n) * size(d)$.
Finally, the *transcoder energy quality TEQ* is determined by subtracting the energy used for the transcoder execution ($E_t$) from the gains in energy caused by transmitting the output data instead of the input data ($E_d - E_{d'}$). The result is further corrected by $E_a$, the energy consumed by the transcoder selection algorithm. Similar to the monitoring of the transcoder run-time, the algorithm monitors its own run-time and uses this to determine the energy overhead of the algorithm. The resulting transcoder energy quality is: $TEQ = (E_d - E_{d'}) - E_t - E_a$.
The resulting number, TEQ, is the energy that can be saved by executing the corresponding transcoder. Only those optional transcoders that have positive TEQs are eligible for data transformation. If more than one transcoder has a positive TEQ, the one with the largest TEQ is applied. If transcoders can be chained, all acceptable transcoder orderings (which can be specified by the client) are considered separately, i.e., TEQs are computed and compared for each suitable chain. In our current implementation, at the receiver side, each transcoder is evaluated for the energy costs of its execution. This approach will be extended in our future work, to also consider the costs of displaying images of different qualities.

**Algorithm Complexity.** If the clock speed is determined by a separate approach, e.g., by a power-aware CPU scheduler, TEQs have to be computed for each available transcoder (and parameter setting) for the given clock frequency and input data size, resulting in $O(t*p)$ operations, where 't' indicates the number of transcoders and 'p' is the number of parameter settings. If the clock frequency can be changed by the transcoder selection algorithm, transcoders have to be evaluated at all clock frequencies. Here, the costs are $O(t * p * n)$, where n is the number of frequency levels. Typically, both the number of transcoders and the number of frequency levels are low (e.g., less than 10 transcoders for video streaming and less than 15 frequency levels for mobile devices).

## 4. Case Study: Video Transcoding

The mobile device under consideration is a Compaq iPAQ H3870 with a StrongARM SA1110 processor. This device supports dynamic frequency scaling and runs a modified version of the *familiar* Linux distribution (version 0.5.2), supporting 11 different clock frequency levels from 59MHZ to 206MHz. The iPAQ consumes about 1.32W of power in idle state (with disabled LCD screen) independent of the used clock frequency, and about 2W when active at the highest clock frequency. A Lucent Technologies Orinoco Gold 11Mbps wireless card is used for the wireless communication. This card causes a power consumption of 0.8W in receive mode and 1.3W in transmit mode. Energy measurements are performed with a Picotech ADC-200 PC Oscilloscope (2 channels, 100kS/second, 12-bit resolution). To facilitate energy measurement, the batteries of both the handheld device and of its extension sleeve are unplugged, thereby forcing the device to draw its power from the DC adapter (5V, 2A max).

**Image Transcoding.** In the following measurements, images are read from disk in PPM format (Portable Pixel Map) and then transformed using a set of image transcoders (gray conversion, crop, reduce, Huffman and LZ77 compression).
The first transcoder, gray, has no input parameters and converts the color coding of an existing image into a gray coding. Figure 4 compares the energy consumption of transmitting unmodified data with the energy consumption of gray converting and transmitting the resulting smaller data. Figure 5 shows a similar experiment for the crop transcoder. Here, we compare the energy consumption for different ratios of output data size to input data size for three different input data sizes. The horizontal line shows the costs of gray conversion of a 518kBytes image in comparison. In Figure 6, we compare the reduce transcoder with three different input data sizes and the costs of gray conversion for data of size 518kBytes. The key result in these graphs is that the energy costs depend both on input data size and the parameters of the transcoder (e.g., the 'reducefactor' for the reduce transcoder). Finally, in Figure 7, we compare the energy consumption for the two compression algorithms. The key result is that two different approaches to the same goal

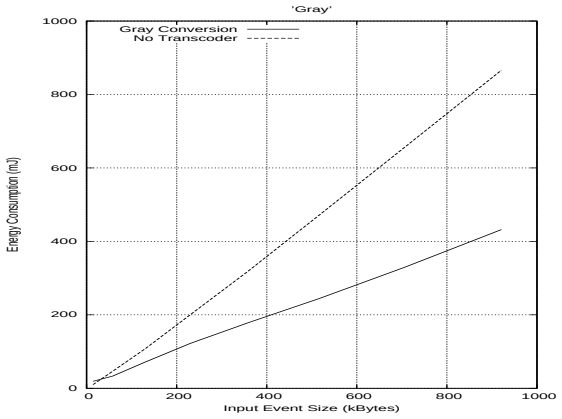


**Figure 4. 'Gray' transcoder.**

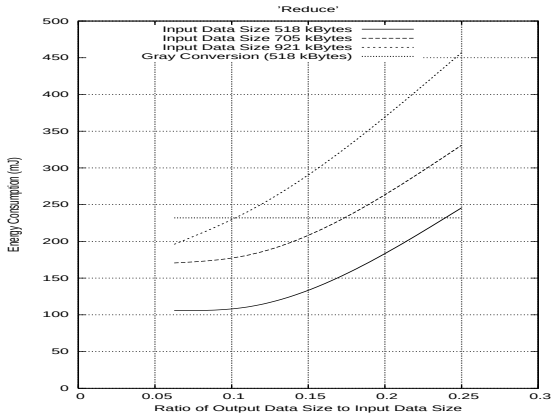


**Figure 6. 'Reduce' transcoder.**

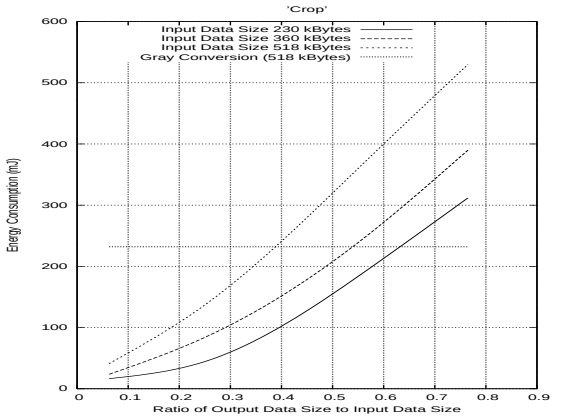


**Figure 5. 'Crop' transcoder.**

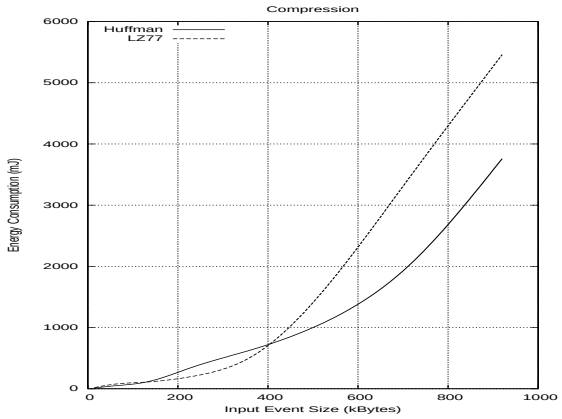


**Figure 7. Huffman and LZ77 compression.**

(compression of data) can have different energy requirements for different data, e.g., data transcoded with the Huffman compression and transmitted over the wireless link requires less energy than the LZ77 compression, except for input data size in the range from 140kBytes to 400kBytes.

**Transcoder Chains.** Transcoders can be chained together, e.g., a client may wish to receive gray-scale images with a certain size, i.e., both the gray transcoder and the crop or reduce transcoders are applicable. An important issue here is the **transcoder ordering**, i.e., the order with which transcoder chains are built. Figure 8 compares combinations of two of the transcoders used in this paper with different orderings, namely the gray and the crop transcoders. The transcoders are first chained such that the gray transcoder is executed before the crop transcoder, then we change the ordering so that the crop transcoder is executed before the gray transcoder. Here, the energy consumption of the crop-gray combination is lower than that of the gray-crop combination. The reason is that the gray transcoder leaves the image size untouched, and the crop transcoder's run-time is independent of the color depth of the image. On the other hand, the crop transcoder reduces the size of the image, thereby also reducing the amount of work required by the gray transcoder (for its pixel by pixel conversion of the image). Next, Figure 9 compares the chains 'gray conversion - LZ77' and 'gray conversion - Huffman', showing how chaining can affect the energy requirements of transcoders. Here, the chain 'gray conversion - Huffman' outperforms (in terms of energy savings) the chain 'gray conversion - LZ77' for all image sizes.

**Data Complexity.** The *content* of data can have an influence on the amount of energy a transcoder function can save. In the case of the gray, crop, and reduce transcoders, image content has no affect on the transcoder run-time – and therefore energy requirements – or output data size (due to the pixel-by-pixel operation of these transcoders). However, the compression algorithms depend on the content of the images. Figure 10 compares the energy consumption

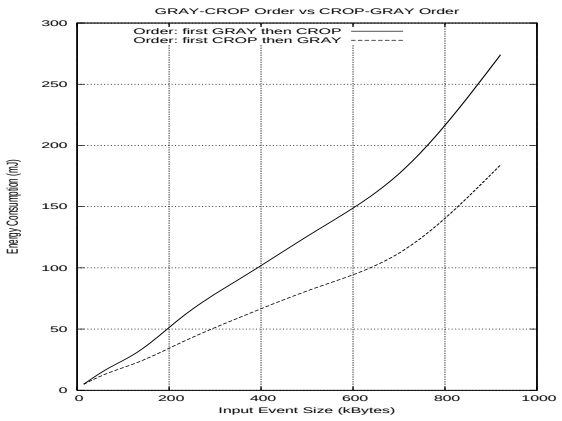


**Figure 8. Ordering: gray and crop.**

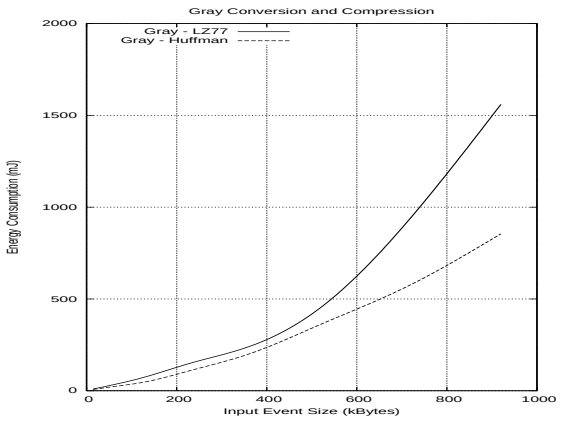


**Figure 9. Ordering: gray and compression.**

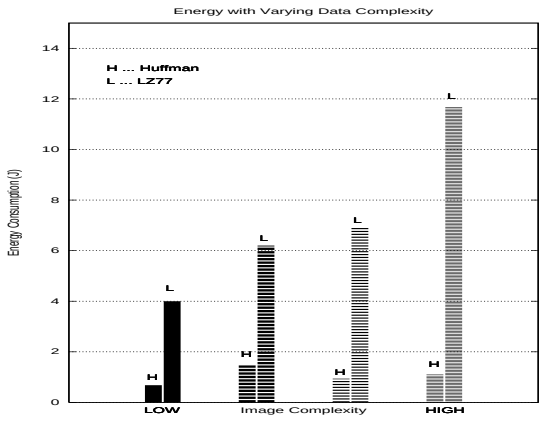


**Figure 10. Variation in data complexity.**

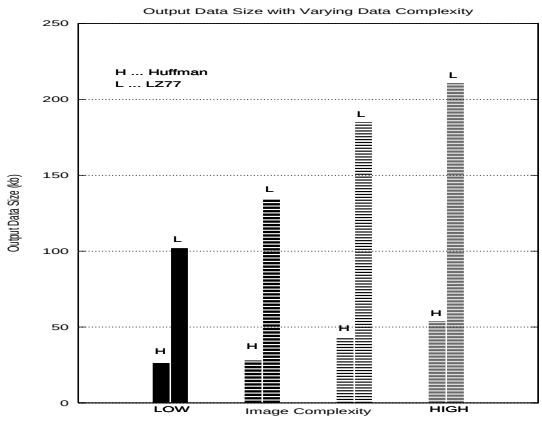


**Figure 11. Variation in data complexity.**

of the Huffman and LZ77 compression algorithms for four images with varying content but identical sizes. Figure 11 compares the size of the transcoded images, which shows that variation in image content has an affect on the achievable image size and therefore the gains achievable with the transmission of smaller-sized data over the wireless link. Both graphs indicate that the content of an image plays an important role in the energy requirements of a transcoding process, which indicates that off-line measurements of the energy characteristics of a transcoder are not sufficient. Although it can be assumed that in many scenarios video image content varies little from image to image, a dynamic approach to determining the characteristics of transcoders is required.

**Power Model.** Power models are used to describe the energy consumption behavior of system devices or software components. We expect that in the near future, 'on-board' power measurement mechanisms will be used to measure the energy consumption of a device, allowing for the automated generation and revision of power models. However, the results presented in this paper rely on obtaining the power model off-line. We are primarily interested in the processing costs of transcoders and the communication costs of submitting media streams. The iPAQ handheld device is examined using an oscilloscope while the transcoders are executed and media streams are transmitted. Figure 12 depicts the measurements of the relationship between the run-time of a transcoder function and the energy requirements caused by the transcoder's execution. Note that this relationship is linear for all clock frequencies for the given architecture. This and all subsequent energy graphs depict the *active* energy, i.e., the total energy minus the idle energy. The results obtained allow us to derive the energy costs for transcoder executions. In comparison to these CPU-centric measurements, it is well-known that wireless network cards consume considerable power for message receipt and transmission. The specification of the Orinoco wireless card notes a receive power of 800mW and a transmit power of 1.3W.

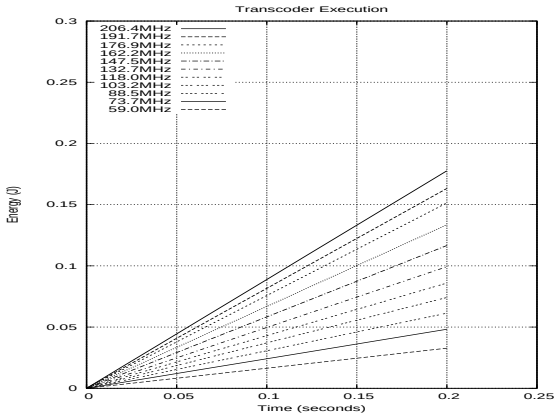


**Figure 12. Energy cost of transcoder execution.**

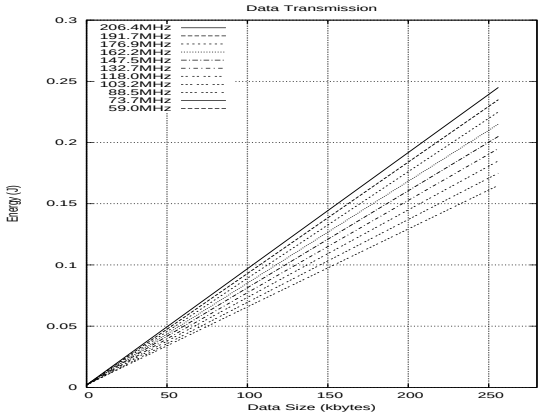


**Figure 13. Energy cost of data transmission.**

For Figure 13, data of varying size is transmitted and the energy consumption for the wireless network card is measured as a function of data size. To utilize the obtained raw energy measurements, the results are turned into factors: $K_r(n)$ for the relationship between transcoder run-time and energy and for a given clock frequency $n$ (Figure 12) and $K_d(n)$ for the relationship between data size and energy (Figure 13).

## 5. Results

The following measurements have been performed with the same setup as in Section 4. In the first experiment, we measure the overheads associated with the transcoder selection mechanism. Figure 14 shows the overheads in microseconds for both the transcoder selection algorithm and the table updates with an increasing number of transcoders. Figure 15 evaluates the sender-side computation of transcoder run-times for the gray, crop, and reduce transcoders for 100 images with random sizes between 1kByte and 900kBytes. The graph shows the number

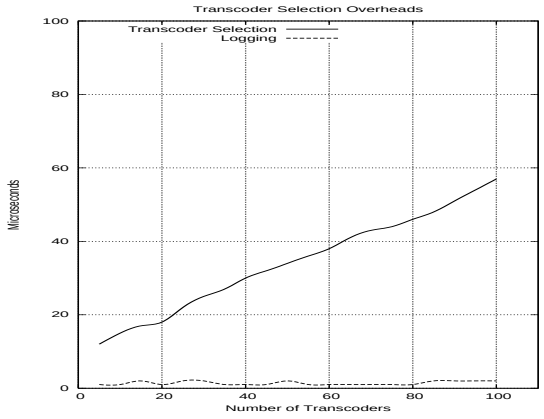


**Figure 14. Overheads.**

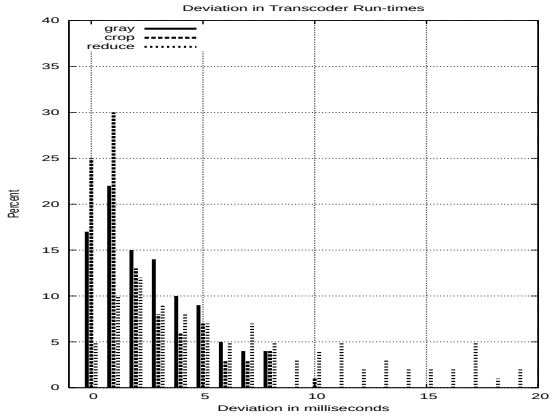


**Figure 15. Deviations of the predicted run-times from the actual run-times.**

of run-time predictions (in %) that deviate from the actual, measured run-times. The accuracy of the run-time predictions determines the accuracy of the energy predictions for the transcoder executions. For example, for an image of size 200kBytes, a deviation in 1ms in run-time prediction results in an error of less than 2% in the energy computation (TEQ) for the gray transcoder. In comparison, the predictions of the output data sizes for the gray, crop, and reduce transcoders were accurate within 1kByte in about 98% of all cases.

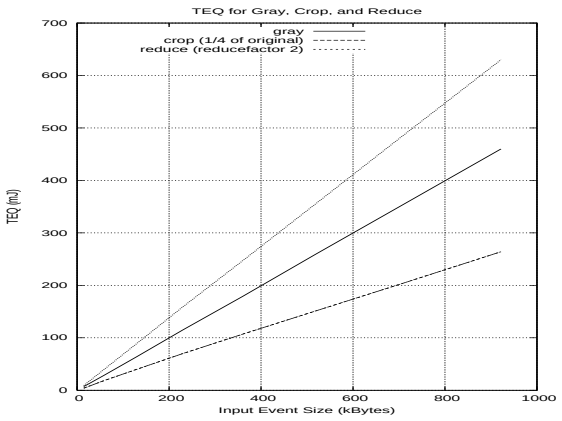


**Figure 16. TEQ Computation.**

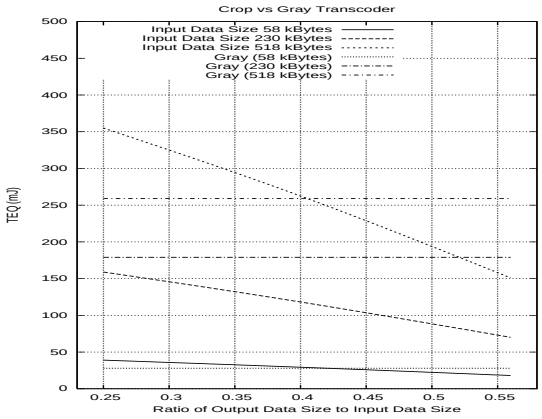


**Figure 17. TEQ Computation.**

Figure 16 and 17 show the results of the TEQ computations for the gray, crop, and reduce transcoders. The first graph compares the three transcoders for different input event sizes, where the parameters for the crop transcoder and the reduce transcoder where set to resize the original image to 1/4 of its original size. The second graph compares the crop transcoder for three different parameter settings with the gray transcoder for data sizes of 58, 230, and 518kBytes. These results indicate that for different parameters and different input data sizes, the TEQ computation will result in different transcoder selections, e.g., for data size of 518kBytes the crop transcoder achieves higher TEQs (and therefore energy savings) than the gray transcoder for images that are cropped to 40% or or less of the original size.

**Discussion.** Table 1 summarizes the mean deviations obtained with our framework. For example, the predictions for the run-times deviate by 3.8ms for the gray transcoder (corresponding to 5.4% of the total run-times), where the predictions for the output data sizes deviate less than 1kByte or 0.2%. These numbers result in a TEQ prediction for the gray transcoder that deviates 3.2mJ (or 5.6%) from the measured results. The predictions for the crop and reduce transcoders are only slightly worse than the ones for the gray transcoder. The approach evaluated here is comparable to the solution

| Transcoder | Run-time | Data Size | TEQ |
|---|---|---|---|
| gray | 3.8ms (5.4%) | 1kByte (0.2%) | 3.2mJ (5.6%) |
| crop | 9.7ms (6.9%) | 1kByte (0.2%) | 7.3mJ (8.4%) |
| reduce | 7.6ms (6.2%) | 1kByte (0.2%) | 6.5mJ (6.2%) |

**Table 1. Mean Deviations**

proposed in [16], where video streams are retrieved from a multimedia server and a proxy between server and clients executes transcoders. Their middleware approach collects residual energy availability information on the client and uses this information on the proxy for real-time transcoding. This is similar to our approach in that relevant energy information of devices is shared and used to perform transcoding, however, our approach addresses situations where both client and server are mobile devices, and transcoder execution is only feasible at either end-host (e.g., in ad-hoc networks, access points without customization capabilities, etc.). Our solution can easily be applied to the same setup as introduced in [16], i.e., where the transcoder selection algorithm is executed in an access point or proxy. Also, our work is closely related to the approach discussed in [13], where encoding and decoding at the server and client is performed in a way to maximize the system lifetime (i.e., the MAX-SOT approach introduced here). This is achieved by exploiting the FGS (Fine Granularity Scalability) video coding technique found in MPEG-4.

## 6. Conclusions and Future Work

Unlike previous approaches, where code off-loading or frequency or voltage scaling is used to reduce the energy requirements of a device, dynamic transcoding introduces additional processing to reduce transmission costs. If data reduction is sufficient, this can be used to reduce the total energy consumed. We address this trade-off in additional processing versus networking costs in the context of a wireless multimedia application, where transcoder functions are used to customize video images. These customizations typically result in smaller data sizes (e.g., down-sampling of video data). We introduce a mechanism that takes into account user preferences for transcoder functions when selecting an appropriate transcoder for transmitted data, with the goal to adhere to global energy management directives. These directives determine how local energy management techniques have to cooperate to obtain and maintain a common global goal.

Our future work will extend the approach introduced here to group communications, i.e., multiple senders and multiple receivers share the media streams and transcoder executions has to be coordinated across the streams. The next version of the transcoder selection approach will be based on a kernel-level quality of service management framework developed by our group. Further, we will extend our experiments to XScale-based handheld devices that feature both frequency and voltage scaling, where the transcoder selection will not only determine suitable transcoders but also determine a suitable clock frequency and voltage for transcoder executions.

## References


[1] M. Anand, E. B. Nightingale, and J. Flinn. Self-Tuning Wireless Network Power Management. In *Proc. of the 9th Intl. Conference on Mobile Computing and Networking*, 2003.

[2] K. Barr and K. Asanovic. Energy Aware Lossless Data Compression. In *Proc. of the First International Conference on Mobile Systems, Applications, and Services*, May 2003.

[3] S. Chandra, C. S. Ellis, and A. Vahdat. Application-Level Differentiated Multimedia Web Services Using Qality Aware Transcoding. *IEEE Journal on Selected Areas in Communications*, 2000.

[4] S. Chandra, C. S. Ellis, and A. Vahdat. Managing the Storage and Battery Resources in an Image Capture Device (Digital Camera) using Dynamic Transcoding. In *Proc. of the Third ACM International Workshop on Wireless Mobile Multimedia (WoWMoM)*, August 2000.

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

[6] J. S. Chase, D. C. Anderson, P. N. Thakar, and A. M. Vahdat. Managing Energy and Server Resources in Hosting Centers. In *Proc. of the 18th ACM Symposium on Operating Systems Principles*, October 2001.

[7] K. Choi, K. Kim, and M. Pedram. Energy-Aware MPEG-4 FGS Streaming. In *Proc. of 40th Design Automation Conference*, June 2003.

[8] R. Cornea, S. Mohapatra, N. Dutt, A. Nicolau, and N. Venkatasubramanian. Managing Cross-Layer Constraints for Interactive Mobile Multimedia. In *Proc. of the IEEE Workshop on Constraint-Aware Embedded Software*, 2003.

[9] R. Cucchiara, C. Grana, and A. Prati. Semantic Video Transcoding for Live Video Server. In *Proc. of ACM Multimedia*, December 2002.

[10] J. Flinn, E. de Lara, M. Satyanarayanan, D. S. Wallach, and W. Zwaenepoel. Reducing the Energy Usage of Office Applications. In *Proc. of the IFIP/ACM Intl. Conference on Distributed Systems Platforms (Middleware)*, November 2001.

[11] J. Flinn and M. Satyanarayanan. Energy-Aware Adaptation for Mobile Applications. In *Proc. of the 17th ACM Symposium on Operating Systems Principles*, December 1999.

[12] C. J. Hughes, J. Srinivasan, and S. V. Adve. Saving Energy with Architectural and Frequency Adaptations for Multimedia Applications. In *Proc. of the 34th Annual Intl. Symposium on Microarchitecture*, December 2001.

[13] A. Iranli, K. Choi, and M. Pedram. Energy-Aware Wireless Video Streaming. In *Proc. of the Workshop on Embedded Systems for Real-Time Multimedia*, October 2003.

[14] Z. Lei and N. D. Georganas. Rate Adaptation Transcoding for Precoded Video Streams. In *Proc. of ACM Multimedia*, December 2002.

[15] Z. Li, C. Wang, and R. Xu. Computation Offloading to Save Energy on Handheld Devices: A Partition Scheme. In *Proc. of CASES'01, Atlanta, GA*, 2001.

[16] S. Mohapatra, R. Cornea, N. Dutt, A. Nicolau, and N. Venkatasubramanian. Integrated Power Management for Video Streaming to Mobile Handheld Devices. In *Proc. of ACM Multimedia*, November 2003.

[17] D. Narayanan, J. Flinn, and M. Satyanarayanan. Uing History to Improve Mobile Application Adaptation. In *Proc. of the 3rd IEEE Workshop on Mobile Computing Systems and Applications*, December 2000.

[18] E. Pinheiro, R. Bianchini, E. Carrera, and T. Heath. Load Balancing and Unbalancing for Power and Performance in Cluster-Based Systems. In *Proc. of the Workshop on Compilers and Operating Systems for Low Power*, 2001.

[19] D. Qiao, S. Choi, A. Jain, and K. G. Shin. MiSer: An Optimal Low-Energy Transmission Strategy for IEEE 801.11a/h. In *Proc. of the ACM/IEEE Intl. Conference on Mobile Computing and Networking*, September 2003.

[20] D. G. Sachs, S. Adve, and D. L. Jones. Cross-Layer Adaptive Video Coding to Reduce Energy on General-Purpose Processors. In *Proc. of the Intl. Conference on Image Processing*, September 2003.

[21] S. Saewong and R. Rajkumar. Practical Voltage-Scaling for Fixed-Priority RT-Systems. In *Proc. of the 9th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, May 2003.

[22] R. Sasanka, C. J. Hughes, and S. V. Adve. Joint Local and Global Hardware Adaptations for Energy. In *Proc. of the 10th Intl. Conference on Architectural Support for Programming Languages and Operating Systems*, October 2002.

[23] A. Vetro and C. W. Chen. Rate-Reduction Transcoding Design for Wireless Video Streaming. In *Proc. of the IEEE 2002 Intl. Conference on Image Processing*, 2002.

[24] W. Yuan and K. Nahrstedt. Process Group Management in Cross-Layer Adaptation. In *Proc. of SPIE/ACM Multimedia Computing and Networking Conference*, January 2004.

[25] W. Yuan, K. Nahrstedt, S. Adve, D. Jones, and R. Kravets. Design and Evaluation of a Cross-Layer Adaptation Framework for Mobile Multimedia Systems. In *Proc. of the SPIE/ACM Multimedia Computing and Networking Conference*, January 2003.