# Application/Architecture Power Co-Optimization for Embedded Systems Powered by Renewable Sources *

Dexin Li and Pai H. Chou
Department of EECS, University of California, Irvine, CA 92697-2625 USA
{dexinl,phchou}@uci.edu

## ABSTRACT

Embedded systems are being built with renewable power sources such as solar cells to replenish the energy of batteries. The renewable power sources have a wide range of efficiency levels that depend on environment parameters and the current drawn from the circuit. Unlike low-power designs whose goal is to minimize energy consumption, systems with renewable power sources should maximize the efficiency of the sources by load matching. To match the wide dynamic range of solar output, it is necessary to exploit multiple power "knobs" simultaneously. This paper combines computation vs. communication trade-offs, algorithm selection, scheduling and dynamic voltage scaling to maximize the dynamic range of the load over time. Experimental results show one to two orders of magnitude performance improvement for a wireless handheld system running image compression applications.

## Categories and Subject Descriptors

C.3 [**Special-purpose and Application-based Systems**]: Real-time and embedded systems

## General Terms

Algorithms, Management, Measurement, Design

## Keywords

power management, renewable power source, power utilization, load matching, architectural optimization

## 1. INTRODUCTION

A growing number of embedded systems is being built to harvest energy such as solar power, wind power, and mechanical vibrations from the environment [1]. Such sources can supply energy indefinitely, but the instantaneous power level tends to vary over a wide dynamic range over time. When the ambient power is available, the system must decide whether to consume or store the power and at what level.

Storing ambient power by charging a battery is convenient and often necessary to enable system operation when ambient power is not available. However, charging is not 100% efficient due to inherent battery characteristics and efficiency loss due to charging circuitry. Alternatively, if it is possible to schedule the workload when ambient power is available, then the same power may be more efficiently utilized. Once the decision is made between battery charging and powering the system, the next question is how much power to draw from the ambient sources. A page-sized solar panel may be rated for 3–5W of output power during peak sunlight, but it does not actually output that much power unless the load is *impedance matched* with the solar panel. Otherwise, only a small fraction of the available power will be output.

Load matching can be accomplished by power management at the architecture level and the application level. At the architecture level, many modern embedded systems support multiple power modes, including operational and idle ones. At the application level, the power manager may switch among different implementations of an algorithm and exploit parameterization for power vs. performance trade-offs. Whether architecture or application level, these are all power "knobs" that can be controlled for the purpose of load impedance matching. However, one problem is that each individual power knob may contribute to a very limited dynamic range of load impedance, whereas renewable energy sources have much wider ranges of output power.

This paper proposes a load matching approach that combines power knobs at multiple levels, including computation vs. communication trade-offs, performance vs. energy trade-offs over multiple implementations of an application, scheduling for maximum resource utility, and dynamic voltage and frequency scaling. The resulting system will have an enhanced dynamic range that will closely match that of ambient power sources. We apply our technique to a solar-powered handheld device that performs image compression and data transmission. Our experimental results show performance improvement by 1–2 orders of magnitude over the straightforward policy. The load matching approach is generally applicable to systems powered by other renewable power sources such as fuel cells [2]. They have similar characteristics to solar panels in terms of a wide dynamic range of output power and nonlinear relationship between the output voltage and the current.

This paper is organized as follows. Section 2 reviews battery-aware and power trade-offs. Section 3 describes aspects of the system modeling. Sections 4 and 5 present the problem statement and our load-matching techniques. Section 6 discusses experimental results.

*This research was sponsored partly by DARPA under contract F33615-02-C-4000 and partly by National Science Foundation under contract CCR-0205712.


*DAC 2005,* June 13–17, 2005, Anaheim, California, USA.

## 2. RELATED WORK

Related work can be divided into power source-aware techniques and system-level power management such as multiple power knob optimization and computation vs. communication trade-offs.

Existing power source-aware research focuses on batteries [3, 4, 5, 6, 7]. The main goal is to increase battery efficiency by shaping the discharge profile to be friendly to the battery by considering the rate capacity and recovery effects. The former is that a higher discharge rate beyond the nominal rate decreases the battery efficiency, while the latter is that the loss efficiency can be recovered if the discharge rate is reduced sufficiently.

While individual low power techniques like dynamic voltage scaling (DVS) are effective in power reduction for subsystems (like a CPU), they are often limited in power optimization that requires global solutions. Yuan et al [8] demonstrated a nice cross-layer framework to coordinate hardware, OS and application layers and to adapt to changes under multiple constraints including quality-of-service, performance, application bandwidth, and power consumption. Mohapatra et al [9] proposed an integrated power management approach that unifies architectural optimizations (CPU, memory, register), OS power-saving mechanisms (DVS), and middleware techniques for client-server-based wireless multimedia applications. StanleyMarbell et al [10] presented algorithms and metrics that characterize system behavior in terms of energy efficiency, reliability, computation performance and battery lifetime for dynamic management in fault-tolerance embedded systems.

Our work addresses the global power optimization issues similar to [8, 9, 10], but differs in the following aspects. First, we take the characteristics of power sources into account in our power optimization. On the contrary, none of the above global optimization work considers the characteristics of power sources. Energy is treated as one of the design constraints regardless of the type of power source it comes from. Second, our goal is to increase dynamic power ranges for best utilization of available energy from power sources, while the goals in the above work are basically trade-offs between multiple design parameters for energy reduction. The energy gain in the above work mainly comes from cross-layer optimization that explores power saving opportunities that would not be possible otherwise. On the other hand, the energy gain in our work comes mainly from improved utility of power sources, which could be additive to multiple power knobs tuning.

System level trade-offs between communication and computation for image compression algorithms were studied in [11, 12]. By tuning algorithm specific parameters, the techniques helped minimize the total energy of computation and communication while satisfying communication bandwidth, image quality and latency constraints. However, the techniques did not further explore the opportunities of combining multiple algorithms together for larger dynamic range of trade-offs.

We developed a measurement-based power model for solar panels in our previous study [13], which can be used to optimize power efficiency of solar panels. In this work we extend the previous work by exploring multiple power knobs for an increased dynamic power range to best match the power output from a solar panel in a handheld system. We incorporate DVS into our approach, together with communication speed selection for improved energy efficiency in both CPUs and communication interfaces. We break the boundary of a single algorithm and allow combined algorithm/parameter selection for image coding applications. Unlike conventional task scheduling approaches that try to utilize the slack time by slowing down the CPU, we propose a scheme in which the slack time is aggressively occupied for more workload in order to best utilize the available power.

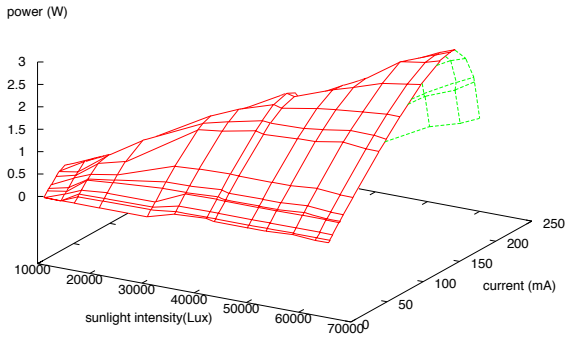


**Figure 1: Measured solar panel characeteristics: output power by sunlight intensity and load current.**

## 3. SYSTEM MODELING

We model an embedded system that has both computation and communication capabilities and is powered by a renewable power source (solar panel). We show the power output of a solar panel is a non-linear function of ambient light intensity and the load current. The application model describes algorithms running on a processor and traffic on a network interface. The resource model depicts both power and performance aspects of architectural resources.

### 3.1 Power Source Model

The impedance of a solar panel is a function of the sunlight intensity and the load. The available models in closed forms [14, 15] describe boundary conditions only and require many parameters that characterize the environment and materials. It is unsuitable for system designers to have a simple model that directly connects solar properties to electrical characteristics. Instead, we developed an empirical model [13] that relates the solar intensity $\phi$, load current $I$, output voltage $V$ and output power $P$, as shown in Fig. 1. The system designer can easily use the model to determine the power budget as a design constraint, as well as the optimal point that outputs maximum power to the load.

### 3.2 Application Model

A task running on a processor can be implemented with one of the algorithms in the set of $\hat{A} = \{a^1, a^2, \ldots, a^v\}$ where $v$ is the total number of alternative algorithms. Each algorithm has up to $l$ configurations of input parameters. We use $a^i_j$ to represent the $i^{th}$ algorithm in the $j^{th}$ configuration. As a matrix,

$$\hat{A} = \begin{pmatrix} a^1_1 & a^1_2 & \ldots & a^1_l \\ a^2_1 & a^2_2 & \ldots & a^2_l \\ \ldots & \ldots & \ldots & \ldots \\ a^v_1 & a^v_2 & \ldots & a^v_l \end{pmatrix} \quad (1)$$

The workload $W$ represents the number of CPU cycles to execute an algorithm. $W(a^i_j)$ is the workload for the $i^{th}$ algorithm with the $j^{th}$ configuration. The data size $U$ is the processed data size for the input to the communication interface. $U(a^i_j)$ is the data size for the $i^{th}$ algorithm with the $j^{th}$ configuration. The quality-of-service $Q$ is the quality factor for the processed data. $Q(a^i_j)$ is quality achievable by the $i^{th}$ algorithm with the $j^{th}$ configuration.

As a case study, we consider a solar-powered handheld device performing three distinct tasks: image acquisition, image compression and data transfer. We assume the image acquisition always consumes constant power and is ignored in the rest of the paper.

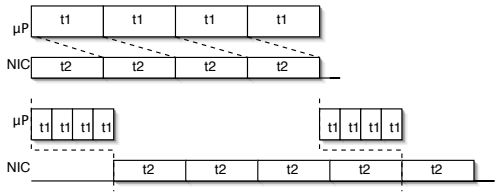


**Figure 2: Above: the processor and the NIC are synchronized ($t_1 = t_2$); below: the processor produces data faster than the NIC can consume ($t_1 < t_2$).**

The image compression can be implemented with one of two alternative algorithms: *jpeg* and *jpeg2000*. Each algorithm has a number of configurations based on different compression ratios. The raw image may also remain uncompressed in this phase. The resulting images are transferred to a remote host through a wireless network interface. The CPU cycles of executing a compression algorithm is denoted as $W$. The compressed image size is denoted as $U$. The image quality is denoted as $Q$, in PSNR (peak signal-to-noise-ratio).

## 3.3 Resource Model

A resource, either a processor or a network interface, has a set of *operational modes* $M_{op}$ as well as a set of *idle* modes $M_{idle}$. Resources are able to process non-zero workload in an operational mode while they cannot process any workload in an idle mode. However the resource in an idle mode consumes less power than it does in an operational mode.

A performance function maps operational modes of a resource to a performance factor $F : M_{op} \rightarrow \mathbb{R}$. It can be the execution speed for a processor or communication bandwidth for a network interface.

The power model of a resource contains a set of power modes $M = M_{op} \cup M_{idle}$, a power function $P : M \rightarrow \mathbb{R}$ that maps power modes to power consumption numbers, and an overhead function $H : M \times M \rightarrow \mathbb{R} \times \mathbb{R}$ that maps mode transitions to time/power pairs.

# 4. PROBLEM STATEMENT

In battery-powered systems, one may slow down or postpone the current work in order to extend battery life; our goal, instead, is to make best use of the available renewable power for more work done. This of course assumes the application allows flexibility to schedule work in the time interval when the ambient power is plentiful. We utilize the slack time and perform extra workload. For a processor, this means compressing more data; for a network interface, this means transferring more data. In fact, the data transferred may include uncompressed data as the result of communication vs. computation trade-offs. We call it *greedy transfer*.

Let $t_1$ and $t_2$ be the delay to compress an image on a processor ($\mu$P) and the time to transfer an image over the network interface (NIC), respectively. The throughput of the system $\psi$ is defined as the number of images transferred in unit time.

If $t_1 = t_2$ (Fig. 2, the upper chart), then the processor and the NIC are synchronized, and $\psi = 1/t_1$. If $t_1 < t_2$ (Fig. 2, the lower chart), then the processor produces data faster than the NIC can consume. The NIC is busy and the processor has slack time that is available for other workload. In this case the throughput $\psi = 1/t_2$.

Let us consider the case when $t_1 > t_2$. A simplistic approach would compress an image, transfer it, compress the next one, and so on (scenario I in Fig. 3). This left much slack time on both the processor and the NIC. A slightly improved approach would use a

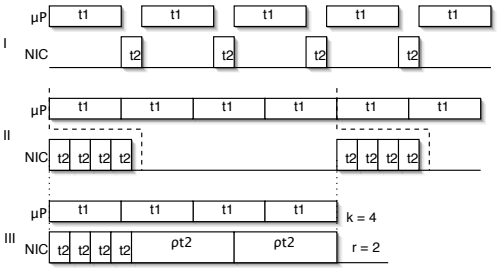


**Figure 3: Three schemes when $t_1 > t_2$. 1) Interleaving data compression with data transfer; 2) compressed data buffered and transfered together; 3) raw data transfered during the idle time on the network interface.**

buffer to store multiple (four in this case) compressed images and transfer them in batch mode, saving network transmission overhead and leaving the NIC longer idle time to set to a low-power mode (scenario II). This approach helps save certain amount of energy but does not improve the system throughput.

With the greedy transfer, we send raw images to the host during the slack time on the network interface. This will not increase the power budget of the system but can improve throughput. This approach assumes the raw images are available whenever we want to utilize the slack time. We also assume one image must be transferred continuously without any interruption, therefore one piece of slack time has to be long enough to accommodate the transfer of at least one raw image. One extra bit is needed to encode whether the transferred information is raw data or compressed data. In practice, this information can be encoded into the file names of the images. The remote host needs an appropriate yet simple decoding function to correctly interpret the received data. In scenario III of Fig. 3, two more images can be transferred within the same amount of time, resulting a throughput improvement of 50%.

In general, given the algorithm and configuration $a_j^i$ of application $\hat{A}$, the selected speeds of processor and NIC as $f$ and $s$, respectively, $t_1$ and $t_2$ can be represented as:

$$t_1 = \frac{W(a_j^i)}{f} \quad (2)$$

$$t_2 = \frac{U(a_j^i)}{s} \quad (3)$$

Let $k$ be the number of processed images in a time period $\tau$ and $r$ be the number of raw data frames that can be transferred in the slack time during the same time period ($k, r \in \mathbb{Z}^+$). $\rho$ is the data compression ratio, which is the ratio between the raw image size and the processed image size. $\rho t_2$ is the time to transfer a raw image. Ideally we have

$$\tau = k \cdot t_1 + k \cdot t_2 + r \cdot \rho t_2 \quad (4)$$

Since $k$ and $r$ are integers, we have

$$\tau = \max(k \cdot t_1, k \cdot t_2 + r \cdot \rho t_2) \quad (5)$$

Given the power functions of the processor and the NIC as $P_{\mu p}(\cdot)$ and $P_{nic}(\cdot)$, respectively, we have the power constraint as

$$P_{\mu p}(f) + P_{nic}(s) \leq P_{max}(\cdot) \quad (6)$$

where $P_{max}(\cdot)$ is the non-linear power constraint imposed by the power source. The system throughput in terms of the number of

```
MULTIKNOBSEL(Â, M_cpu, M_nic, q, P_max, τ, u)
    ▷ Input:  the matrix of algorithms and parameters Â
    ▷         a set of modes of processor (frequencies) M_cpu
    ▷         a set of modes of NIC (transfer speeds) M_nic
    ▷         the peak signal-to-noise ratio q
    ▷         the maximum power constraint P_max
    ▷         the time period τ
    ▷         the raw data size u
    ▷ Output: a ∈ Â, f ∈ M_cpu, s ∈ M_nic, ψ
 1  M_c ← ∅                                   ▷ subset of M_cpu that satisfies p_max
 2  M_n ← ∅                                   ▷ subset of M_nic that satisfies p_max
 3  M_cn ← ∅                                  ▷ mode combinations that satisfy p_max
 4  Φ ← ∅                                     ▷ a set of throughput with mode combination
 5  for each f ∈ M_cpu
 6      for each s ∈ M_nic
 7          p' ← P_cpu(f) + P_nic(s)
 8          if p' ≤ P_max, then M_cn.append(p', f, s)
 9  if M_cn is empty, then return NIL
10  A' ← ∅
11  for each a ∈ Â
12      if Q(a) ≥ q
13          A'.append(a)
14  if A' is empty, then return NIL
15  for each a ∈ A'
16      for each c = (p', f, s) ∈ M_cn
17          t1 ← W(a)/f; t2 ← U(a)/s
18          k ← ⌊τ/t1⌋
19          ρ ← u/U(a)                        ▷ image compression ratio ρ
20          if t1 > t2
21              r ← ⌊τ − k·(t1 − t2)/ρt2⌋
22              ψ ← (k + r)/max(kt1, t2(k + rρ))
23          else
24              ψ ← ⌊τ/t2⌋/τ
25          Φ[a][c] ← (a, f, s, ψ)
26  return max(Φ)                             ▷ return the mode combination with max throughput
```

**Figure 4: Combined algorithm selection and resource speed selection for maximum throughput.**

images transferred is

$$\psi = \frac{k+r}{\tau} \tag{7}$$

Our goal is to maximize the system throughput $\psi$ subject to the power constraint (6) by selecting $f$, $s$, and $a_j^i$.

# 5. ALGORITHM

Given a maximum power constraint $P_{max}$, an image quality constraint $q$ and a raw image size $u$, MULTIKNOBSEL (see Fig. 4) determines the algorithm and its configuration $a_j^i$, the CPU speed $f$ and the NIC speed $s$ that gives maximum throughput.

As a pre-processing step, MULTIKNOBSEL prunes the search space by eliminating infeasible mode combinations that violate either of the constraints (line 5–14). In the main search loop (line 15–25), the algorithm checks whether $t_1 > t_2$ for each of mode combinations. If true, it lets the NIC sends raw image during the slack time (line 21–22); otherwise the NIC sends processed data as fast as possible (line 24).

The algorithm MULTIKNOBSEL runs in polynomial time bounded by $|\hat{A}||M_{cpu}||M_{nic}|$. Usually $\hat{A}$, $M_{cpu}$, $M_{nic}$ are small and the pre-processing step can prune the search space effectively. In practice, it is even feasible to use the algorithm for on-line purposes.

# 6. EXPERIMENTAL RESULTS

We test our techniques on a solar-powered handheld device with a wireless network interface. In this section we first describe the experiment setup and then present results.

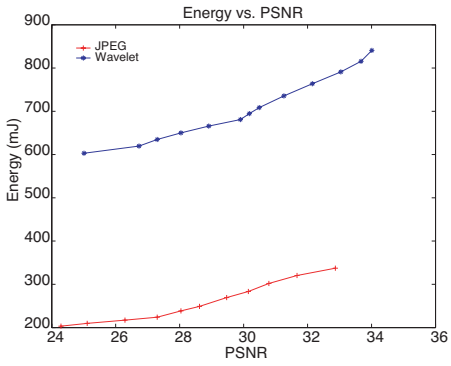


**Figure 5: Combined algorithm/parameter selection for larger dynamic energy range. Each algorithm has a dynamic range from 30% to 50%. When combined, they provide a dynamic range of up to 400%.**

| Speed | Power |
|---|---|
| 59MHz | 320mW |
| 89MHz | 598mW |
| 118MHz | 866mW |
| 142MHz | 1088mW |
| 167MHz | 1319mW |
| 206MHz | 1680mW |

| Speed | Bandwidth | Power |
|---|---|---|
| 11Mbps | 5200kbps | 1825mW |
| 5.5Mbps | 4300kbps | 1532mW |
| 2Mbps | 1800kbps | 1282mW |
| 1Mbps | 920kbps | 1215mW |

**Figure 6: Power modes of the CPU and the NIC.**

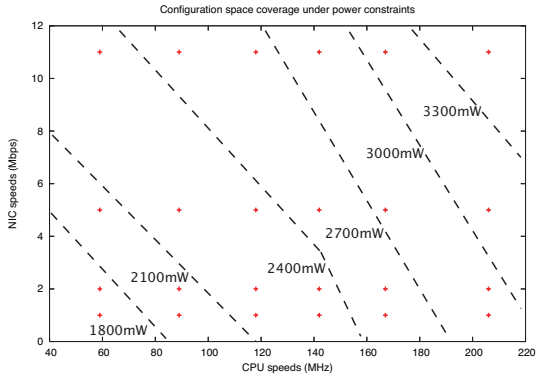


**Figure 7: Coverage of mode combinations by power constraints.**

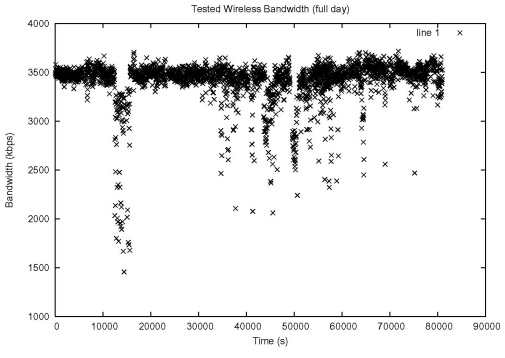


**Figure 8: Measured NIC bandwidth during the time of a day.**

## 6.1 Experiment Setup

We studied two image compression algorithms, namely *jpeg* and *jpeg2000*. *Jpeg* runs faster and consumes less energy than *jpeg2000* does. However the compressed image size with *jpeg* is usually 30–70% larger than that with *jpeg2000* for the same image quality. Each algorithm provides a dynamic range about 30–50% in energy consumption while the two combined provide a dynamic range of up to 400% and about 300% in image sizes (not shown).

We apply our techniques to a solar-powered (Siemens ST5, through a switching regulator National LM2675) handheld iPaq 3850 with a wireless NIC Cisco Aironet 350. To guarantee the repetitiveness and predictability of our experiments, we did not use a real solar panel directly exposed to the Sun for our tests. Instead, we use our self-developed solar panel emulator [13] as the power source. The handheld continuously processes an image (lena 512x512) and sends out the processed images over the wireless NIC. We use a profile of wireless bandwidth from field measurements (Fig. 8). We use the solar panel model shown earlier (Fig. 1) to derive the maximum power constraints. Power modes of the handheld and the network interface are shown in Fig. 6.

At the runtime, the handheld examines the sunlight intensity using the built-in light sensor and determines the power budget of the solar panel. This routine runs every 60 seconds ($\tau = 60.0$), and the power modes of CPU and NIC are updated accordingly.

## 6.2 Results

We first show how well the combination of power modes can adapt to different levels of power budget, followed by the throughput improvement under various power constraints and image quality requirements. We will also show results from a test that compares several schemes including a battery-powered one. Lastly, we give the execution efficiency of our technique.

### *Power Mode Coverge*

The solar panel we used can deliver up to 3.5*W* to the load. Fig. 7 shows the coverage of power modes under different maximum power constraints. The x-axis and y-axis represent power modes of the processor and the NIC, respectively. In a wide range of power budget ($1.5 - 3.5W$), we can choose combinations of power modes to let the system work. The higher the power budget is, the more mode combinations we can choose from.

### *Throughput Improvement*

We test the system throughput under different image quality and maximum power constraints (see Fig. 9). The dark bar in each column represents the throughput without the greedy transfer. The light bar represents the *additional* throughput gain from our technique. When the requirement for the image quality is relatively low (Fig. 9(a)), the *jpeg* algorithm is selected most of the time due to its light-weight computation and high energy efficiency. As a result, more images can be transferred than when the system requires a higher image quality (Fig. 9(b)). In the latter case, *jpeg2000* is likely to be chosen. Due to the higher compression ratio of *jpeg2000* (marked *J2k* in the figure), the delay for transferring compressed images is shortened, leaving more time for transferring raw images.

In real life conditions, the processor and NIC may not be able to fully utilized all the time. For example, the CPU may be shared by multiple applications and the network bandwidth is shared by multiple users. In fact, each resource may be allocated a portion of its total utility. We vary the CPU utilization and NIC utilization and can determine the throughput improvement at certain power image quality constraints. The result is shown in Fig. 11. Each point in the mesh represents the mode combination that gives highest throughput under the given utility rates. In all cases we achieve more than 100% throughput improvement. Specially, when the processor is less utilized than the NIC, we achieve much higher throughput rate. This is because we can choose to send more raw images using the relatively faster NIC.

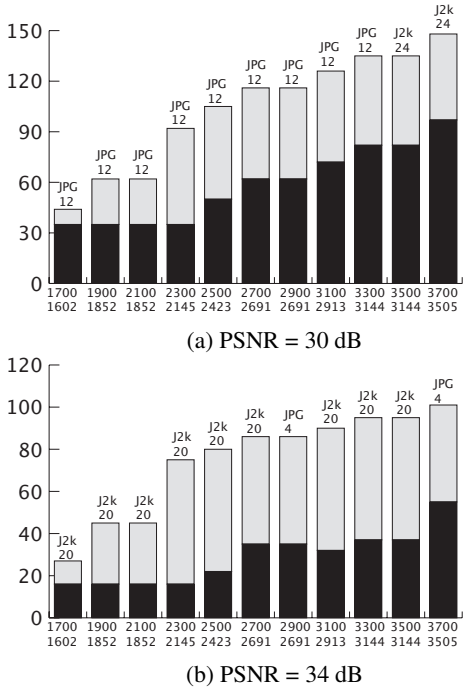


**Figure 9: Throughput under maximum power constraint. Dark bars: the number of compressed images; light bars: the number of raw images transferred. Numbers on the x-axis: power constraints (top) and actual power consumption numbers (bottom).**

| | On time | | Energy | | Throughput | |
|---|---|---|---|---|---|---|
| Method | (Hour) | (%) | (mAHr) | (%) | (#) | (%) |
| Full-on | 2.14 | 100% | 1440 | 100% | 3436 | 100% |
| Simple LM | 6.22 | 290% | 3337 | 232% | 7369 | 214% |
| Advanced LM | 6.22 | 290% | 5004 | 347% | 10621 | 309% |
| Battery | 0.58 | 27% | 391 | 27.1% | 784 | 22.8% |

**Figure 10: Comparison of different power management schemes. Full-on: set to mode M0 all the time; conservative: set to mode M5 all the time.**

### *Comparision of Field Tests*

We test three scenarios using the same sunlight profile and bandwidth profile from real life. In the first two scenarios, we use the solar panel emulator as the only power source. In the *full-on* scenario the resources are set to the power modes with the maximum performance. The simple load matching scenario (*simple LM*) does not support the algorithm selection and the greedy transfer, and the computation and communication are performed sequentially (case I in Fig. 3). The advanced load matching (*advanced LM*) incorporates all the techniques discussed above, including power mode selection, algorithm/configuration selection, and greedy transfer. In the last scenario, we first fully charge a rechargeable battery

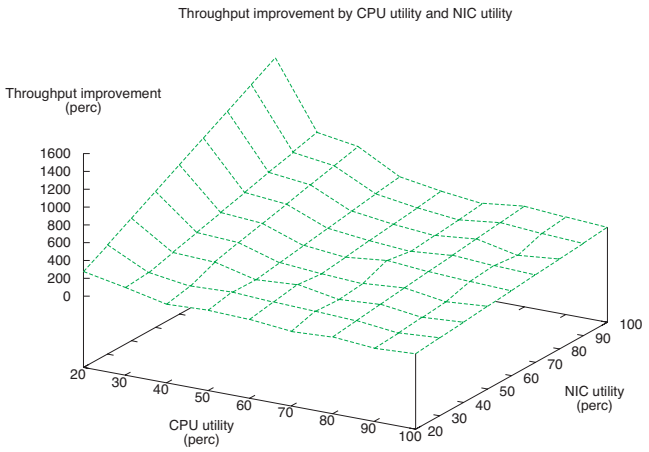


**Figure 11: Throughput improvement (in percentage) by CPU utilization and NIC utilization.** $P_{max} = 2500$**, PSNR**$= 32$**.**

| Algorithm | Run time | Percentage |
|---|---|---|
| JPEG, $\rho = 12$ | 381ms | 100% |
| MODESELECTION | 8.2ms | 2.2% |

**Figure 12: Execution time of JPEG and** MULTIKNOBSEL**. CPU speed set to 206MHz.PSNR =**32**dB**

(NiMH, 1600mAH, 600mAH/hr) using the same solar panel emulator, and then use the battery as the only power source to power the system. Based on the given sunlight profile, we can fully charge the battery only once during the time of a day (see Fig. 10).

With load matching, both *simple LM* and *advanced LM* achieve 2*X* and 3*X* throughput improvement over the *full-on* approach. This is because both methods are able to keep track of available power and maintain longer on-time for more work done. Our techniques achieve about 50% improvement over *simple LM* mainly because we fill the slacks on the NIC with additional workload.

Moreover, our techniques achieve 1–2 orders of magnitude higher throughput over the battery-powered approach. This is reasonable because: 1) our approach avoids efficiency loss during energy conversion, and 2) our approach does not suffer from non-ideal effects of batteries (the rate capacity effect and the recovery effect).

Last, our algorithm MULTIKNOBSEL executes very fast compared to the execution time of compressing even one single image (see Fig. 12).

# 7. CONCLUSION

Utilizing available power from natural power sources in distributed embedded networks presents both challenges and opportunities for system designers. We take a first step by proposing a system-level technique that makes best use of the power for throughput improvement. This is done by performing load matching with combined algorithm selection for an application and power mode selection for resources. The goal is to keep the resources busy while still satisfying the power constraints. Our technique shows quite significant throughput improvement over battery-powered systems. We envision that the life time of a mix-powered embedded network could be greatly improved with load balancing between battery-powered nodes and solar-powered nodes. To fulfill that, our load matching is an enabling technique.

# 8. REFERENCES


[1] T. Sterken, K. Baert, R. Puers, and S. Borghs. Power extraction from ambient vibration. In *Proc. of WorkshoponSemiconductor Sensors*, pages 680–683, November 2002.

[2] U. Pasaogullari and C.Y. Wang. Computational fluid dynamics modeling of solid oxide fuel cells. In *Proc. of Solid Oxide Fuel Cells VIII, Eds. S.C. Singhal and M. Dokiya*, pages 1403–1412, 2003.

[3] K. Lahiri, S. Dey, and A. Raghunathan. Communication architecture based power management for battery efficient system design. In *Proc. of Design Automation Conference*, pages 991–996, June 2002.

[4] J. Luo and N. K. Jha. Battery-aware static scheduling for distributed real-time embedded systems. In *Proc. of Design Automation Conference*, pages 444–449, June 2001.

[5] S. Park and M. B. Srivastava. Dynamic battery state aware approaches for improving battery utilization. In *Proc. of Compilers, Architectures and Synthesis for Embedded Systems*, pages 225–231, 2002.

[6] D. Bruni, L. Benini, and B. Ricc. System lifetime extension by battery management: an experimental work. In *Proc. of Compilers, Architectures and Synthesis for Embedded Systems*, pages 232–237, 2002.

[7] D. Rakhmatov, S. Vrudhula, and D.A. Wallach. A model for battery lifetime analysis for organizing applications on a pocket computer. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 11:1019–1030, December 2003.

[8] W. Yuana, K. Nahrstedta, S. V. Advea, D. L. Jonesb, and R. H. Kravetsa. Design and evaluation of a cross-layer adaptation framework for mobile multimedia systems. In *Proc. of the SPIE/ACM Conference on Multimedia Computing and Networking*, January 2003.

[9] S. Mohapatra, R. Cornea, N. Dutt, A. Nicolau, and N. Venkatasubramanian. Integrated power management for video streaming to mobile handsets. In *Proc. of ACM Multimedia*, November 2003.

[10] P. Stanley-Marbell and D. Marculescu. Dynamic fault-tolerance and metrics for battery powered, failure-prone systems. In *in Proc. IEEE/ACM Intl. Conference on Computer-Aided Design (ICCAD)*, November 2003.

[11] N. Taylor and S.Dey. Adaptive image compression for enabling mobile multimedia communicatiodesigon. In *Proc. IEEE Intl. Conference on Communications*, pages 1925–1929, 2001.

[12] D.G. Lee and S. Dey. Adaptive and energy efficient wavelet image compression for mobile multimedia data services. In *Proc. of Conference on Communications*, pages 2484–2490, 2002.

[13] D. Li and P.H. Chou. Maximizing efficiency of solar-powered systems by load matching. In *Proc. International Symposium on Low Power Electronic Design (ISLPED)*, pages 162–167, August 2004.

[14] D.L King, J.A. Kratochvil, and W.E. Boyson. Field experience with a new performance characterization procedure for photovoltaic arrays. In *Proc. 2nd World Conference and Exhibition on Photovoltaic Solar Energy Conversion*, 1998.

[15] M.W. Davis, A. Hunter Fanney, and B.P. Dougherty. Measured versus predicted performance of building integrated photovoltaics. *Journal of Solar Energy Engineering-Transactions of the ASME*, 125:21–7, 2003.