Ultra-Low Power Data Storage for Sensor Networks *

Gaurav Mathur, Peter Desnoyers, Deepak Ganesan, Prashant Shenoy
{gmathur, pjd, dganesan, shenoy}@cs.umass.edu
Department of Computer Science
University of Massachusetts, Amherst, MA 01003

ABSTRACT

Local storage is required in many sensor network applications, both for archival of detailed event information, as well as to overcome sensor platform memory constraints. While extensive measurement studies have been performed to highlight the trade-off between computation and communication in sensor networks, the role of storage has received little attention. The storage subsystems on currently available sensor platforms have not exploited technology trends, and consequently the energy cost of storage on these platforms is as high as that of communication. Current flash memories, however, offer a low-priced, high-capacity and extremely energy-efficient storage solution.

In this paper, we perform a comprehensive evaluation of the active and sleep-mode energy consumption of available flash-based storage options for sensor platforms. Our results demonstrate more than a 100-fold decrease in per-byte energy consumption for surface-mount parallel NAND flash in comparison with the MicaZ on-board serial flash. In addition, this dramatically reduces storage energy costs relative to communication, introducing a new dimension in traditional computation vs communication trade-offs. Our results have significant ramifications on the design of sensor platforms as well as on the energy consumption of sensing applications. We quantify the potential energy gains for two commonly used sensor network services: communication and in-network data aggregation. Our measurements show significant improvements in each service: 50-fold and up to 10-fold reductions in energy for communication and data aggregation respectively.

Categories and Subject Descriptors

B.7.1 [**Integrated Circuits**]: Hardware Types and Design Styles Memory Technologies; B.8.2 [**Hardware**]: Performance and Reliability Performance Analysis and Design Aids

General Terms

Measurement, Performance, Experimentation

Keywords

Sensor Networks, Embedded Systems, Storage, Flash Memory, Energy Efficiency

1. INTRODUCTION

Wireless sensor networks has been an area of significant research in recent years. Sensor deployments are often untethered, and their energy resources need to be optimized to ensure long lifetime. A significant fraction of sensor network research has addressed the problem of energy-efficiency, primarily by exploiting the fact that computation is many orders of magnitude less expensive than radio communication. This *computation vs communication trade-off* has had a tremendous influence both on algorithm design as well as on sensor network platform design. For instance, each new generation of sensor platform (e.g. the Berkeley mote) has involved a transition to a more energy-efficient and capable processor and radio.

While the evolution of sensor platforms has tracked technology trends in computation and communication components, the storage subsystem across these platforms has undergone little change. All generations of the Mica motes provide limited storage of a less than a megabyte, at an energy cost equivalent to or greater than that of communication. The high energy cost of storage has raised questions about the rationale for using in-network storage-based data management techniques for sensor networks. If storage indeed consumes as much energy as communication, most sensor networks should be designed to be centralized data-collection systems, and in-network storage techniques will have limited applicability. Conversely, if storage uses significantly less energy than communication, greater emphasis should be placed on exploiting storage to minimize communication and hence conserve energy.

To understand these trade-offs, we ask: *What is the most energy-efficient storage platform for sensor networks, and what are its implications on sensor network design?* We address this question in two parts. First, we perform an exhaustive empirical characterization of the active and sleep-mode energy consumption of flash-based storage options to determine the most energy-efficient storage device for sensor

*This work is supported by a grant from the Engineering Research Centers program of the National Science Foundation under cooperative agreement EEC-0313747 and NSF grants CNS-0546177, CNS-052072 and EIA-0080119.

IPSN'06, April 19–21, 2006, Nashville, Tennessee, USA.

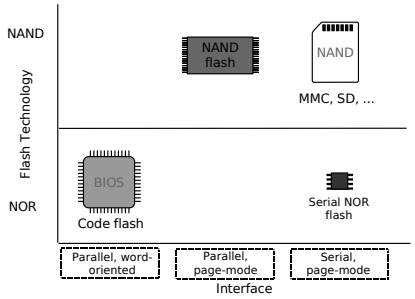


Figure 1: Flash Technologies

networks. Our results show that parallel NAND flash technology offers a 100-times more energy-efficient storage option compared to other flash memories and the radio on the MicaZ [5]. This challenges conventional wisdom that considers the energy cost of storage to be equivalent or greater than that of communication.

The second part of our research evaluates the impact of high-capacity, ultra-low power storage on sensor network design. We measure the impact of flash memory trends on two services: communication and in-network data aggregation. Our measurements show a 50-fold reduction in energy consumption for communication and up-to 10-fold additional reduction for data aggregation.

2. BACKGROUND

Semiconductor-based flash memory is now widely used in applications ranging from BIOS code on motherboards to image storage in digital cameras. With recent improvements in capacity and power, flash memory is now a viable storage technology for low-power, energy-constrained wireless sensor network devices. While there are other compact or low-power storage technologies such as micro-drives, flash is the only one which meets sensor network storage requirements of low energy consumption, ultra-low idle current, and high capacity.

Flash memory falls into several categories, illustrated in Figure 1. The most fundamental difference is in the underlying memory cell technology; flash is classified as NOR or NAND flash according to the type of the circuit that holds a single bit. NOR flash is less dense than NAND, and uses more energy for erase and programming, but provides random read access times of less than 100ns. NAND flash allows sequential read only and has significantly higher starting latency, but has the ability to stream subsequently read bytes at high speed. NAND flash cells are also less reliable than NOR cells, requiring the use of error correcting codes (ECC) for reliability.

Flash devices may be *word* or *page-oriented* internally. A word-oriented device behaves like static RAM - it receives an address on an address bus, and outputs a data word on the data bus. A page-oriented device operates like a disk drive - commands are sent identifying a block of data, and a response is returned. Similar to disks, write operations (and often reads) operate on an entire page. NOR flash is available in either organization, while NAND flash, being inherently sequential, is only available in page-oriented form.

Type	Manufacturer	Capacity	Interface	Page size	Erase block (pages)
Serial NOR	Atmel	512 KB	SPI	256B	1
Serial NOR	ST	512 KB	SPI	256B	256
MMC	Hitachi	32 MB	SPI	512B	16
NAND flash	Toshiba	16 MB	8-bit bus	512B	32
NAND flash	Micron	512 MB	8-bit bus	2048B	64

Table 1: Characteristics of tested devices

Flash memory is available as surface-mount components for circuit assembly, or as standardized removable devices such as MultiMedia Cards (MMC) [19] or SecureDigital (SD) [27] cards. Interfaces to flash devices are either *parallel*, transfering an entire byte or word at a time, or *serial*, transfering bits one by one. NOR flash chips are available with either interface, while current NAND chips are only available with parallel interfaces; serial-interface removable devices such as MMCs must translate the parallel NAND interface to serial.

A common characteristic of all flash technologies is that writes are "one-way" - i.e. after writing a memory location, it must be reset or *erased* before it may be written again. This operation is relatively slow and expensive, and must be performed on one or more fixed-sized regions known as *erase blocks*. In some devices the erase block size is the same as the write block size (e.g. one page), but frequently erase blocks span multiple pages, complicating flash memory management.

Removable flash devices expose a standardized set of disk-like operations that are implemented by an internal microcontroller, which in turn translates them into operations on the NAND interface (see Figure 2). This controller performs operations such as erasure, page remapping, ECC, and wear leveling[1], simplifying system design while increasing power consumption due to the additional internal circuitry. The use of surface-mount NAND devices eliminates this overhead, but requires flash memory management to be performed either in software or using additional hardware.

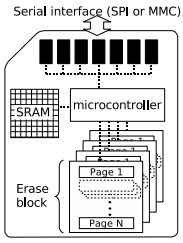
3. EVALUATION OF FLASH DEVICES

3.1 Devices Tested

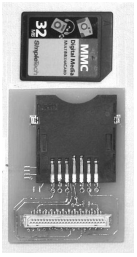
We examine the energy consumption of three different currently available flash memories that are appropriate for sensor networks: serial NOR flash, removable NAND flash with controller, as represented by the MultiMedia Card (MMC) format, and byte-parallel surface-mount NAND flash. Although byte-parallel NAND flash is available in removable form as xD cards [1], we were unable to include them in our study as the manufacturer has not made the technical specifications publicly available.

Serial NOR: The Atmel AT45DB041B [3] 512kByte serial NOR flash was chosen due to its extensive use as the external data flash on the Mica series motes. We also test the STM25P80 flash on the TelosB [28]. Figure 3 shows both these devices. Both motes were modified to allow measurement of current consumption by the serial flash devices.

[1]A single page of flash memory may only be erased and written a certain number of times - typically 10^5 - before degrading; to maximize lifetime, write operations must be distributed over the entire device.



Organization



Mica2 MMC adapter

Figure 2: Multimedia Card (MMC). The adapter shares the SPI bus with the Mica2.

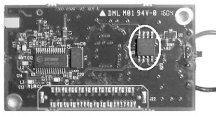
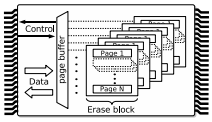
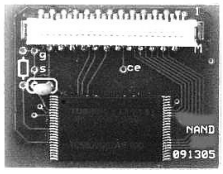


Figure 3: Atmel AT45B041 serial NOR flash on Mica2 mote.



NAND flash organization



NAND flash adapter

Figure 4: The NAND flash adapter board for the Mica2 - it shares the expansion connector with other sensors.

	Energy per byte (μJ)				
	Read	Write	Erase	Bulk Erase (Page count)	Total
Atmel NOR	0.26	4.3	2.36	n/a	6.92
Telos NOR	0.056	0.127	n/a	0.185 (256)	0.368
Hitachi MMC	0.06	0.575	0.47	0.0033 (16)	1.108
Toshiba 16MB NAND	0.004	0.009	n/a	0.004 (32)	0.017
Micron 512MB NAND	0.027	0.034	n/a	0.001 (64)	0.062

Table 2: Flash Energy Consumption - Read, Write, and Erase

MMC: An MMC adapter was designed and fabricated for the Mica series of motes (Figure 2) and TinyOS [15] drivers written to access the MMC in Serial Peripheral Interface (SPI) mode. We tested four different makes of MMC devices and report the results of the best performing device, the Hitachi HB28D032MM2 [13].

NAND Flash: We designed and built a parallel NAND flash board (Figure 4) and TinyOS drivers for the Mica mote[2]. We tested the following devices: Toshiba TC58DVM-72A1 16MB device [30] and Micron MT29F4G08BAB 512MB device [17].

The tested devices are enumerated in Table 1 along with a summary of their characteristics.

3.2 Methodology

We measure both the *active mode* and *sleep mode* power consumption of each flash device. The active mode measurements consist of power consumption when performing read, write or erase operations. The sleep mode measurements indicate the current drawn by the device in its lowest power consumption mode.

Measurement of all devices was performed on a Mica2 mote. The current was measured at the device using power leads with a 10Ω sense resistor and a digital oscilloscope, and traces of each operation were integrated to give total energy consumption. The need for a high degree of accuracy required the use of a precision multimeter to measure sleep mode current. The mote was powered by an external power supply with a supply voltage of 3.3V; energy consumption "in the field" with a partially discharged battery may be somewhat lower.

[2]Hardware design and device drivers are available on our web site, [http://sensors.cs.umass.edu/projects/storage]

Note that under some circumstances, MMC write performance may degrade as the device fills, due to internal fragmentation [26]. In order to avoid this effect, the MMC devices were fully erased before write testing.

3.3 Read, Write and Erase Energy Usage

Read and write energy costs of single pages were measured, and these are presented in Table 2. The results are presented in units of energy per byte, to account for the difference in page and erase block sizes of the devices.

The energy cost for read, write and erase operations on the Telos NOR is seen to be 18x less than the Atmel NOR and 3x less than the Hitachi MMC. The Toshiba 16MB NAND flash is 21x more efficient in comparison to the Telos NOR, 65x better than the Hitachi MMC and 407x better than the Atmel NOR. The Micron 512MB NAND flash was found to be 3.6x less efficient than the Toshiba 16MB NAND flash, though offering 32 times the storage capacity. Many factors affect the energy consumption of flash memories, but we are unable to discuss these due to the space constraints of this paper. We consider the Toshiba 16MB NAND flash for further discussions.

The results of the erase operation may also be seen in Table 2; the minimum erase block size was tested for the serial NOR and the NAND devices - 1 and 32 pages respectively. The MMC interface defines both a single page erase and a block erase command; both were tested. We find that erasing a single page is 140 times more expensive than a block erase of several 16-page blocks.

Under continuous usage, one byte must be erased for every byte written. Thus, in this case the total energy used to write a single byte should also consider the erase operation that precedes it. For the MMC, the per-byte energy cost of erasure was negligible compared to writing - 0.0033μJ vs. 0.575μJ. With the Toshiba 16MB NAND flash, erase required 0.004μJ/byte, which added to a write energy cost of 0.009μJ/byte gives a total of 0.013μJ while the Micron 512MB NAND flash consumes a total of only 0.035μJ/byte, marginally higher than the write cost. In either case, accounting for erase energy does not significantly increase the total energy cost.

The energy consumed by each read and write operation has two components - a constant overhead associated with the operation and a variable component that is dependant on the size of data being read or written. Figure 5 shows how the energy consumption varies with varying data sizes on the NAND flash, and this is representative of the energy

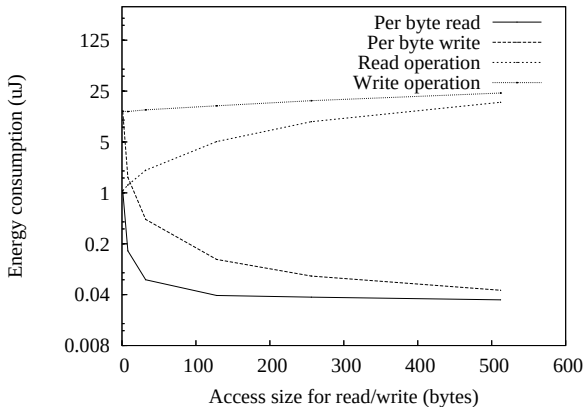


Figure 5: Affect of size of data being read or written on the energy consumption.

	Sleep current μA	Power-up μJ
Serial NOR	2	0
Hitachi MMC	84	1130
Toshiba 16MB NAND	5	0

Table 3: Sleep Mode current and Power-on energy consumption

consumption on most NAND and NOR devices. We see that the overhead of the read operation is fairly small, thus the read storage sub-system can afford to have small or no data buffering. In contrast, the write operation has a significant overhead and having a page write buffer amortizes the fixed cost over a larger number of data bytes, reducing the per byte write cost.

3.4 Idle Current, Startup and Shutdown Power Consumption

The idle power consumption of a device is important for an energy-limited platform. If the sensor node only uses the flash infrequently, the idle current may dominate both active and sleep mode energy consumption and thus determine battery and system lifetime. Table 3 shows the measured idle current for the three flash devices, and compares it with the vendor specification. The NOR and NAND devices draw very little current - 2μA and 5μA respectively. These values are comparable to the idle current of the mote CPU, at between 5μA and 15μA [2], or the self-discharge current of an alkaline AA cell, which is in the range of 10μA[8]. In contrast, the idle current for the MMC is 17 times greater, which could significantly impact maximum system lifetime.

Idle current drain may be eliminated by powering off the device when idle; however, when powering up again, MMCs incur significant energy costs for initialization of the internal controller on power-up. Table 3 shows the startup energy consumption for the Hitachi MMC; at 1130μJ, power-up consumes about as much energy as writing 2000 bytes or reading 20,000 bytes.

3.5 ECC and CPU Energy Consumption

In most applications a NAND flash must be used with an error correcting code capable of correcting single bit errors.

	Energy per byte (μJ)				
	Data Transfer	Read		Write	
		Without ECC	With ECC	Without ECC	With ECC
Hitachi MMC	0.084	0.06	0.144	0.575	0.659
NAND flash	0.011	0.004	0.030	0.009	0.034

Table 4: Total Energy Consumption for Flash and CPU

When used in a system without hardware support for ECC, the function must be performed in software. To determine its energy cost, we measured the performance of a 512 by 8 block parity implementation [25] on the Mica2 mote. The measured energy values for the read and write operations are adjusted to account for the additional ECC cost in Table 4.

The energy cost of ECC calculation on the Mote is quite significant at 0.015μJ/byte and is almost 4 times the energy consumption of the flash. A more efficient CPU like the 16-bit MSP-430 used on the Telos can compute the same ECC twice as fast consuming half as much energy. Special-purpose hardware may be able to reduce the overhead even further, as ECC is a simple function to compute in hardware.

Table 4 also identifies the energy cost due to the CPU performing data transfer in software. This is necessary on the Mica, as the only high speed interface, SPI, is not exposed on the expansion connector. Transferring data from memory to the NAND flash consumes 0.011μJ per byte, which is almost 3 times that consumed by the device itself. Our optimized software SPI driver for the MMC requires 4 cycles for each bit transfered, expending 0.084μJ per byte.

Hardware support for data transfer might significantly reduce the overheads associated with ECC - access to a hardware SPI port would double the serial transfer rate and a DMA controller such as is found on the MSP-430 would allow the CPU to sleep during data transfer, reducing overall energy consumption.

3.6 Summary and Discussion

Our measurements show that parallel NAND flash is the most energy efficient storage device for sensor networks. Although the MMC is also based on NAND technology, the presence of an internal micro-controller increases idle current as well as energy consumption for read, write and erase operations. Additionally, the byte-wide interface of parallel NAND proves to be significantly faster and more efficient than bit-serial ones. However, the comparatively larger number of I/O pins required for the parallel interface may pose interfacing issues on low power embedded systems with limited number of I/O pins. An ideal storage solution for sensor networks should combine the performance of the parallel NAND flash with the lower pin count of serial interfaces.

In order to better exploit the lower energy cost of NAND flash, further platform-level optimizations need to be performed. We observe that the current energy cost of performing ECC in software and keeping the CPU active during data transfer from flash to memory is greater than the device energy consumption. Thus, any further significant reduction in the storage cost requires re-design of other platform components with the storage subsystem in mind.

	MSP-430 instruction	Toshiba NAND Read	Toshiba NAND Write	CC2420 Radio Tx	CC2420 Radio Rx
Energy (μJ/byte)	.0008*	0.004	0.009	1.8	2.1
Ratio	1	5	11	2250	2600

*Calculated from measurements in [22].

Table 5: Per-byte energy usage: Communication, Storage, and Computation

4. IMPLICATIONS ON SENSOR SYSTEMS

The low energy consumption of NAND flash memory motivates a more extensive investigation of the energy cost of storage, in comparison to computation and communication in sensor network systems. We compare our measurements of the Toshiba NAND flash with vendor-specified values for the Chipcon CC2420 radio [4] and the TI MSP-430 CPU [29], used on the Telos [22] motes. The energy numbers of the Toshiba 128MB NAND flash does not affect the obtained results.

Table 5 shows the energy comparison for these devices. We notice that the energy used in reading a byte from NAND flash storage is only 5 times that used for a "unit" of computation - i.e. touching a byte in RAM. Additionally, writing a byte to flash is only 11 times as expensive as computation. In contrast, radio transmission represents a 200-fold increase in energy usage over writing to NAND flash, and reception a 500-fold increase over reading.

In the spectrum of energy costs, we see a small gap between computation and storage, and a large separation between storage and communication. Current research in wireless sensor systems has focused on the trade-off between computation and communication, ignoring the role of storage. Our results significantly change conventional wisdom about the relative energy costs of these operations, and enable algorithms to trade expensive communication for cheaper storage.

The low energy consumption and falling price of flash memory will soon make it possible to equip each wireless sensor node with a few gigabytes of NAND flash memory. This translates to cheap and virtually unlimited storage for the sensor: in many cases every sensor reading, every packet transmitted, and every packet received during the sensor lifetime can be logged to storage. To put this in perspective, consider a seismic monitoring applicaiton that is optimized to last for two years running on the MicaZ. If this application were to generate as much as 512 bytes of data per second, and store every single byte to NAND flash storage, the lifetime of each sensor would reduce by only 6 weeks, having stored a total of 28GB of data.

This high-capacity, energy-efficient flash memory is valuable to applications not only for persistent long-term data archival, but also to overcome RAM limitations on sensor platforms. We briefly describe some applications that can benefit from such storage.

In-network Query Processing: While much research has addressed the topic of in-network storage and querying, local storage has not been a viable alternative to communication on currently available sensor platforms. We speculate that this one of the reasons why existing sensor network deployments have relied primarily on centralized data collection techniques for query processing and have not exploited in-network storage.

The availability of ultra-low power, high-capacity NAND flash memory for local storage offers new advantages to in-network storage and querying mechanisms, where the sensors use the flash to locally archive sensed data. Instead of centralized query processing, local data archives can be effectively exploited to retrieve data from sensors only when a query requests the data, thereby conserving energy.

Use of History: A number of sensor applications maintain local models for adapting to changes in their environment. In many cases, these models are constructed using time-series of past data collected from the environment. For example, predictive storage mechanisms such as PRESTO [7] use data history to build time-series models of temperature data. Harvesting-aware power management schemes [23] rely on the history of harvested energy to predict the energy that can be harvested at a particular time in the future. The presence of cheap storage allows utilization of more history data to develop more accurate models resulting in greater energy savings.

Network-level compression: In-network data aggregation schemes such as Directed Diffusion [12] rely on hash tables to perform duplicate packet suppression. These hash tables are often too large for RAM and need to be constructed on flash storage. Flash-based data management schemes such as MicroHash [31] could be used to store these hash tables. NAND flash enables us to construct larger hash tables at lower energy cost, thereby improving the performance of such schemes.

Custody Transfer: Delay Tolerant Networks (DTNs) [9] have been studied extensively in recent research literature. Routing in a DTN relies extensively on custody transfer, wherein a large amount of "batched" data is transferred one hop closer to its destination each time two nodes meet.

The lack of sufficient storage at sensor nodes makes the use of DTN concepts infeasible for sensor networks. However, in light of our results, NAND flash motivates similar custody-transfer based routing techniques to enhance duty cycling, leading to longer sensor lifetimes.

5. RE-THINKING SENSOR NET DESIGN

This section examines possible approaches to exploit ultra-low power parallel NAND flash and quantifies the reduction in energy expenditure accompanying each of these approaches. We examine the energy reduction for services that emphasize different dimensions in sensor network design – communication, data processing and sensing, and highlight the implications of flash storage on each of them. The first of these is discussed in Section 5.2 and shows the impact of flash storage on *communication costs* when duty cycling is used. Section 5.3 measures the effect of storage on *data processing* costs by analyzing in-network data aggregation. We consider the impact on this set of services to be representative of the tremendous impact that flash based cheap storage can have on sensor systems.

5.1 Experiment Methodology

Sensor network services typically involve three operations - computation, storage and communication. In a periodic sensing application, each of these operations may be characterized by two parameters: the *frequency* and the *magnitude* of the operation. The magnitude could be the number

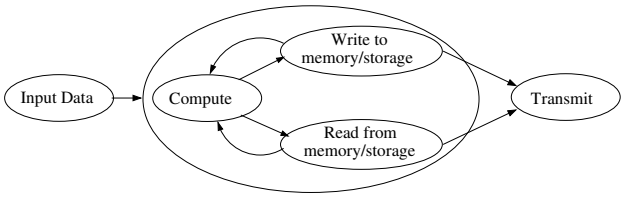


Figure 6: Sensor network service model. Processing proceeds from input to transmit, with specified (possibly zero) amounts of work at each stage.

of processing cycles executed, the number of bytes stored or retrieved, or the number of bytes transmitted or received. In our model we assume the period and magnitude for each operation to be constant during the application lifetime. These parameters thus characterize the energy consumption of a sensor network service.

We developed a *sensor service emulator* that models a typical sensor service as depicted in Figure 6. The emulator takes the magnitude and frequency parameters discussed above as input and executes the actual operations on the MicaZ sensor platform, thus expending equivalent energy. An external power supply of 3.3V was connected to the battery terminals, and total system current was measured with a digital oscilloscope and 10Ω dropping resistor. As we consider large data sizes in our study, the TinyOS packet size was increased from 29 bytes to 128 bytes. We use the emulator to model the services discussed in subsequent sections.

5.2 Impact on Communication Service

Our first experiment examines the reduction in energy costs of communication that can be achieved with energy-efficient local storage. Typical sensor nodes expend significant energy on data transmission and reception, as the active state power consumption of the radio is quite high: approximately 20mA in either transmit or receive mode. Thus, the radio is turned on only when required and kept in sleep mode otherwise. The number of times the radio is power cycled also needs to be minimized as there is a fixed energy cost associated with radio startup and shutdown. The availability of efficient local storage allows the application design to utilize simple batching mechanisms to amortizes radio startup and shutdown energy costs over a larger number of data bytes. Although such batching increase the latency of data collection, in most sensing applications this will not be a concern.

Sensor networks extensively use *duty cycling* - a sensor node puts itself to sleep and only wakes up periodically to listen, conserving energy. These protocols often incur significantly higher per packet transmission energy costs, however, as the transmitter must do more work to ensure the receiver wakes up and receives the packet. We consider the BMAC[3] [21] protocol, where decreases in the receiver duty cycle in turn require corresponding increases in the length of the packet preamble transmitted by the sender. This adds a fixed overhead for a preamble to the per packet transmission

[3]The CC2420 does not support BMAC currently, but this is expected in the near future. For our study, we assume the size of the packet preamble to be the same as that of the CC1000 and use the CC2420 energy numbers and transmission speed to calculate the energy cost of transmission.

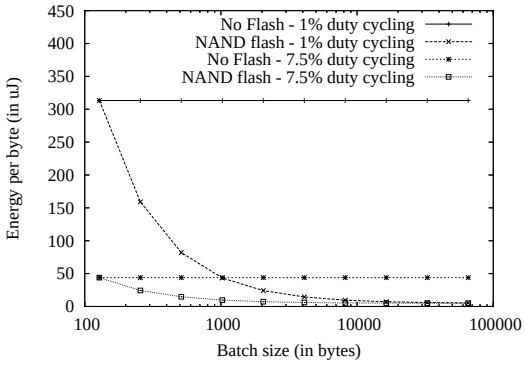


Figure 7: Energy expenditure: batching with and without external NAND flash. Duty cycles of 1% and 7.5% are shown.

energy cost, which is also inversely proportional to the duty cycle of the sensor.

Figure 7 shows the effect of duty cycling on the energy consumption of two applications - one with no extra storage, which buffers up to one packet of data in RAM before transmission, and one which performs simple batching using the NAND flash and transmits data in batches. The effect of two duty cycling rates, 1% and 7.5%, is illustrated here. We notice that the application using NAND flash storage provides significant energy gains for any batch size greater than 128 bytes. In the 1% duty cycling case, the flash-enabled application consumes 3.8 times less energy/byte even for a small batch size of 512 bytes or 4 packets. This decreases further with increasing batch sizes, achieving a 58-fold improvement for a batch size of 65536 bytes or 512 packets. With 7.5% duty cycling the packet preamble is smaller, reducing the per packet fixed energy cost.

We conclude that using local storage to perform batching in duty-cycled applications reduces communication energy costs (up to 58x) in comparison to a non-batching approach, by amortizing the per-packet transmission overhead over a larger number of data bytes.

5.3 Impact on Data Aggregation

The volume of data generated by a sensor network is often high, requiring some level of in-network data aggregation and compression to reduce the number of transmitted bytes. High-capacity energy-efficient local storage allows larger amounts of data to be accumulated and compressed at once, providing more efficient compression leading to lower transmission costs. Our goal is to study the impact of flash storage, the computational complexity of the compression scheme and its compression ratio on the energy consumption of the application.

We model three classes of in-network data aggregation schemes in our study. Certain applications require the captured sensor data in its entirety and use a *lossless encoding* scheme like arithmetic or Huffman encoding [24]. Lossless encoding schemes are often computationally expensive and yield compression ratios that increase slowly as the amount of data compressed increases. Other applications are more willing to tolerate loss of data granularity for higher compression ratios, and use *lossy encoding* schemes. These

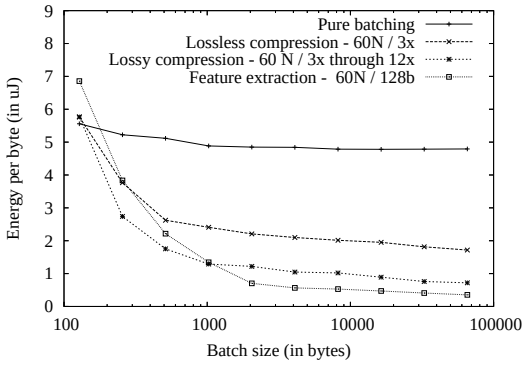


Figure 8: Energy usage of compression schemes using NAND flash with: lossless compression, lossy compression, feature extraction and no compression. Computational complexity is assumed to be $60N$ in each case, with 100% duty cycling.

schemes trade data accuracy for lower computational complexity and/or higher compression ratios, which typically increase with increasing data sizes. Some sensor applications perform *feature extraction* or event detection, where a relatively low complexity algorithm is employed which scans the data generated by the signal transformation phase, looking for certain thresholds or characteristics. The output of these algorithms is typically independent of the input data size and a certain number of output bytes is generated per detected feature vector.

We consider a data collection application where the collected sensor data is stored on the flash. A batching task executes periodically that aggregates the data using one of the compression schemes described above and transmits the result to the data sink. We do not instantiate specific instances of compression algorithms in our study, but consider computational requirements and compression ratio as sufficient parameters to represent the different classes of compression algorithms.

We examine the impact of different compression schemes on the net energy expenditure. Our benchmark of a wavelet compression scheme [10] optimized for sensor platforms (no floating point operations) yields a computational complexity of $60N$, where N is the input data size. Since this experiment aims to show the difference between various compression schemes, we consider $60N$ as the common computational complexity for all the schemes. Next, we assume that the lossless compression scheme yields a constant compression ratio of 1:3, and the lossy scheme yields a compression ratio that starts at 1:3 for 128 bytes and increases by a factor of 1 each time the batch size doubles. The feature extraction scheme is assumed to yield a constant output of 128 bytes for all input data sizes. We also include the energy costs obtained for simple batch processing in Section 5.2 for reference.

Figure 8 shows the results obtained for the experiment. We observe that the benefits of using any compression scheme over simple batching become apparent for any batch size greater than 128 bytes. For a batch size of 512 bytes, the batching scheme without compression consumes twice the energy of any of the compression-enabled applications. As expected, the lossy compression scheme yields better energy per byte results than the lossless scheme, since the compression ratio increases with batch size. The feature extraction scheme transmits 128 bytes irrespective of the input batch size and turns out to be more expensive for smaller batch sizes, whereas the other compression schemes compress to fewer than 128 bytes. The feature extraction scheme becomes cheaper than the lossless scheme for batch sizes greater than 300 bytes and becomes the most efficient one for all batch sizes greater than 1000 bytes. The energy gains of the compression scheme increase for large batch sizes, and for a batch size of 65536 bytes we see a 10-fold reduction in energy consumption for batching with compression in comparison to pure batching.

We conclude that significant energy gains can be obtained by using NAND flash based local storage for performing in-network data aggregation (up to 10x in our experimental setup), in comparison to a pure batching approach.

6. RELATED WORK

To our knowledge, no published work to date compares different alternatives for energy-efficient local flash storage in sensor networks, or the changes in storage, computation and communication trade-offs that emerge as energy costs of storage decrease sharply. A few studies have quantified the energy consumption of individual flash memory devices chosen as part of sensor platform designs. The RISE project [18] at UC Riverside has developed a new sensor platform with an interface to external SD/MMC card flash storage. They present measurements [31] of energy consumption for a single SD card, which are comparable to those for one of the less-efficient MMC cards we tested. Our NAND flash adapter uses 10 times less energy for read and 150 times less energy for write than this device.

From a systems perspective, several studies such as DIMENSIONS [10] and PRESTO [7] make use of local storage, but work on these projects has focused on the applications and systems themselves. A number of filesystems have been proposed and implemented for sensor nodes, including Matchbox [11] and ELF [6], but these focus on organizing flash into a filesystem for data storage and retrieval, rather than the actual energy cost of the flash storage itself.

Several studies have quantified the energy consumption of currently available flash storage on motes - accurate energy figures for the Mica flash are presented as part of the energy budget planning for the Great Duck Island deployment [16], and power consumption data is available for the flash storage on the Telos mote [22]. Other studies have compared energy use and performance of flash storage technologies in the context of hand-held battery-powered devices [20, 14]. The results of these studies are not directly applicable to sensor platforms as typical hand-held device focus more on the performance than on ultra-low power consumption, since the latter is a secondary concern given the use of rechargeable batteries in personal devices.

7. CONCLUSION

This paper identifies parallel NAND flash as the most energy efficient storage device for sensor networks. Our measurement study shows it to be 100-fold more energy efficient than the serial NOR flash present on the Mica platform. The significant energy reduction offered by parallel

NAND flash motivates a re-examination of the computation-communication trade-off, to include a careful analysis of the storage dimension as well. We observe energy costs for storage to be two orders of magnitude less than for communication, significantly changing conventional wisdom that considered storage costs to be comparable to those of communication.

This has significant implications for sensor network design and we evaluate the impact on three commonly used sensor network services – communication, in-network data aggregation, and localization and quantify the energy reduction achieved. Our measurements show at least an order of magnitude reduction in energy costs for the communication and data aggregation services and improved localization accuracy at low additional energy cost. Our results make a compelling case for the use of parallel NAND flash based storage subsystems for sensor network platforms.

8. REFERENCES

[1] *xD-Picture Card*. `www.xd-picture.com`.

[2] Atmel Corporation. *ATMega128 Datasheet*, Nov. 2004.

[3] Atmel Inc., `www.atmel.com/`. *4-megabit 2.5-volt or 2.7-volt DataFlash AT45DB041B*, 2005.

[4] Chipcon. CC2420 2.4 GHz IEEE 802.15.4 / ZigBee-ready RF transceiver, 2004.

[5] Crossbow Technology, Inc., San Jose, CA. *MPR/MIB User's Manual*, Sep 2005.

[6] H. Dai, M. Neufeld, and R. Han. ELF: an efficient log-structured flash file system for micro sensor nodes. In *SenSys '04: Proceedings of the 2nd International conference on Embedded networked Sensor systems*, pages 176–187, New York, 2004. ACM Press.

[7] P. Desnoyers, D. Ganesan, H. Li, and P. Shenoy. PRESTO: A predictive storage architecture for sensor networks. In *Tenth Workshop on Hot Topics in Operating Systems (HotOS X).*, June 2005.

[8] Duracell, `www.duracell.com/OEM`. *Alkaline Mangenese Dioxide Technical Bulletin*.

[9] K. Fall, W. Hong, and S. Madden. Custody transfer for reliable delivery in delay tolerant networks. Technical Report IRB-TR-03-030, Intel, July 2003.

[10] D. Ganesan, B. Greenstein, D. Perelyubskiy, D. Estrin, and J. Heidemann. An evaluation of multi-resolution search and storage in resource-constrained sensor networks. In *Proc. of the 1st ACM Conference on Embedded Networked Sensor Systems (SenSys)*, Nov 2003.

[11] D. Gay. Design of matchbox, the simple filing system for motes. in TinyOS 1.x distribution, `www.tinyos.net`, August 21 2003. Version 1.0.

[12] J. Heidemann, F. Silva, C. Intanagonwiwat, R. Govindan, D. Estrin, and D. Ganesan. Building efficient wireless sensor networks with low-level naming. In *Proceedings of the Symposium on Operating Systems Principles*, pages 146–159, Banff, Alberta, Canada, October 2001. ACM.

[13] Hitachi, Ltd., `www.hitachi.com`. *HB28E016MM2 / HB28D032MM2 HB28D064MM2 / HB28B128MM2 MultiMediaCard*, 2002.

[14] H. G. Lee and N. Chang. Energy-aware memory allocation in heterogeneous non-volatile memory systems. In *ISLPED '03: Proceedings of the 2003 International Symposium on Low Power Electronics and Design*, pages 420–423, 2003.

[15] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler. TinyOS: An operating system for wireless sensor networks. In *Ambient Intelligence*. Springer-Verlag, 2005.

[16] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson. Wireless sensor networks for habitat monitoring. In *ACM Intl. Workshop on Wireless Sensor Networks and Applications*, Atlanta, Sept 2002.

[17] Micron Technology, Inc., `www.micron.com`. *Datasheet: MT29F4G08BABWP / MT29F4G16BABWP*.

[18] A. Mitra, A. Banerjee, W. Najjar, D. Zeinalipour-Yazti, D. Gunopulos, and V. Kalogeraki. High performance, low power sensor platforms featuring gigabyte scale storage. In *Third International Workshop on Measurement, Modeling, and Performance Analysis of Wireless Sensor Networks (SenMetrics 2005)*, San Diego, CA, Jul 2005.

[19] MMCA Technical Committee, `www.mmca.org`. *The MultiMediaCard, Based on system specification version 4.1*, 2005.

[20] C. Park, J.-U. Kang, S.-Y. Park, and J.-S. Kim. Energy-aware demand paging on NAND flash-based embedded storages. In *ISLPED '04: Proceedings of the 2004 international symposium on Low power electronics and design*, pages 338–343, 2004.

[21] J. Polastre, J. Hill, and D. Culler. Versatile low power media access for wireless sensor networks. In *Proceedings of the Second ACM Conference on Embedded Networked Sensor Systems (SenSys)*, November 2004.

[22] J. Polastre, R. Szewczyk, and D. Culler. Telos: Enabling ultra-low power wireless research. In *Proc. 4th Intnl. Conference on Information Processing in Sensor Networks: Special track on Platform Tools and Design Methods (IPSN/SPOTS)*, April 2005.

[23] M. H. Rahimi, H. Shah, G. S. Sukhatme, J. S. Heidemann, and D. Estrin. Studying the feasibility of energy harvesting in a mobile sensor network. In *Proc. of the 2003 IEEE Intl. Conference on Robotics and Automation(ICRA)*, pages 19–24, 2003.

[24] D. Salomon. *Data Compression: The Complete Reference*. Springer Verlag, second edition, 2000.

[25] Samsung Electronics Co., Ltd., `www.samsung.com`. *ECC Algorithm (512 Byte)*, 2005.

[26] SanDisk. *White Paper: Sandisk flash memory cards wear leveling*, Oct. 2003.

[27] SD Card Association. *SD Memory Card Architecture*.

[28] STMicroelectronics, `www.st.com`. *Datasheet: M25P80*.

[29] Texas Instruments. *MSP430x1xx Family User's Guide*.

[30] Toshiba America Electronic Components, Inc. (TAEC), `www.toshiba.com/taec`. *Datasheet: TC58DVM72A1FT00*, Jan. 2003.

[31] D. Zeinalipour-Yazti, S. Lin, V. Kalogeraki, D. Gunopulos, and W. Najjar. MicroHash: An efficient index structure for flash-based sensor devices. In *4th USENIX Conf. on Files and Storage Technologies (FAST 2005)*, San Francisco, CA, December 2005.