NODES: A NOVEL SYSTEM DESIGN FOR EMBEDDED SENSOR NETWORKS

S. Neema, A. Mitra, A. Banerjee, W. Najjar, D. Zeinalipour-Yazti, D. Gunopulos, V. Kalogeraki
University of California - Riverside
{sneema, amitra, anirban, najjar, zeinalipour, dg, vana}@cs.ucr.edu

ABSTRACT

Emerging technologies provide increasingly powerful, efficient, compact and economically viable capabilities like single chip solutions for wireless embedded sensor systems, and large capacity flash memories. In this paper we present the RISE (RIverside SEnsor) platform, a novel system design for embedded sensors built around a System-on-Chip device interfaced with a large external storage memory in the form of off-the-shelf SD (Secure Digital) Card. RISE supports a new paradigm of "sense and store" as opposed to the prevalent "sense and send" for sensor networks.

We describe the hardware and software structure of RISE which supports the standard TinyOS and NesC environment. We demonstrate that significant energy savings together with the additional benefits of reduced complexity and increased ease of use are achieved by adopting the sense and store methodology in which we transmit only the data of interest. It has been determined that percentage energy savings per node for storing data as against transmitting over a single hop can be expressed as $(92.2 - 105.9x)$, where x is the fraction of useful data that needs to be transmitted. Also investigated, is a number of applications that benefit from the extra degree of freedom afforded by a large storage media.

1. INTRODUCTION

The design of wireless sensor systems owe their origins to the dictates of harsh resource constraints in size, cost, energy and memory. To this date sensors consist of assembled discrete components and small amounts of memory [8]. However, it has become apparent that an increasing number of applications require a wide spectrum of capabilities on a platform and require more resources than others [12]. Some applications may even require nodes in the same network to perform different functions. Trends in memory prices and size indicate that storage memory will continue to get cheaper and denser, and will thus cease to be a critical economic constraint. Flash memory allows for a very low energy storage budget. Also, higher levels of device integration at low cost and size now provide us with powerful, compact and economic single chip solutions for our sensory and communication needs. The RISE (RIverside SEnsor) platform leverages these trends essentially by employing the Chipcon™ CC1010 device. The CC1010 is a true single-chip RF microprocessor / transceiver with an integrated high performance 8051 microcontroller that benefits from a very wide array of publicly available software including compilers and debuggers.

Our contributions include porting of the most prevalent design environment, the TinyOS (version1.1) and NesC (version 1.2alpha1), facilitating easier and modular programming, interfacing of an SD-Card, and a carbon dioxide sensor and developing the reactive methodology of query based response on large datasets stored locally on the nodes. We are currently developing a simple file system to support efficient access to the memory. The open source SDCC (small device C compiler) is used to generate the 8051 compatible C code. The platform lowers communication costs significantly by reducing the overall traffic to include only the subset of the data that is of interest to an application querying the node. We calculated that the energy cost of writing to flash memory is less than 10% of the RF transmission cost. As an example, if the relevant data is only a half of the data sensed, power consumption is reduced by more than 35%. Other benefits include sensory interleaving between multiple sensor nodes, on-chip indexing and reduced complexity of the design. The RISE platform will be initially deployed to measure carbon-dioxide levels in soil and monitor bird populations by sensing and processing ambient sound.

The rest of the paper is organized as follows: Section 2 describes the motivation and requirements of the RISE platform. The RISE platform, its components and the design decisions are described in Section 3. Section 4 details the sense and store paradigm. In Section 5 we evaluate the energy gains accrued by adopting this methodology. Section 6 discusses the benefits to applications from this methodology. We conclude with a survey of related work (Section 7) and a discussion of NODES and future directions (Section 8).

2. MOTIVATION

Our work is motivated by the requirements of the Bio-Complexity and the James Reserve projects at the Center of Conservation Biology (CCB) at UC Riverside [3]. CCB is working towards the conservation and restoration of species and ecosystems by collecting and evaluating scientific information. The bio-complexity project is designed to develop the kinds of instruments that can monitor the soil environment directly in environments where factors like high humidity and precipitation will be challenge for the sensors, rather than in laboratory recreations**.** One of the goals is to improve understandings of the spatial and temporal processes that control soil carbon sequestration in a tropical seasonal forest and the role of soil microorganisms. The objectives in particular are to study soil carbon in a fire chronosequence to evaluate an ecological restoration experiment in terms of carbon and to integrate spatially and temporally the information from the sensor arrays with ecosystem scale measurements (e.g. root biomass, litter, soil carbon).

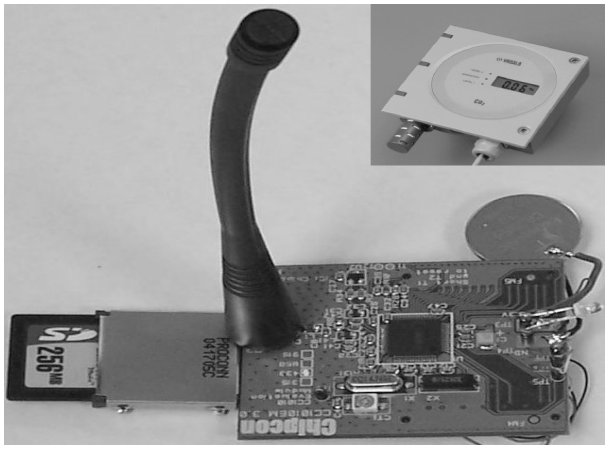

Fig1. RISE Platform, Inset (Vaisala CARBOCAP GMT)

3. THE RISE SENSOR PLATFORM

The RISE node shown in Figure 1 is based on a modular design using widely available commercial off-the-shelf technology. RISE was designed from the ground up, to serve applications demanding more flexibility, and performance rather than power thriftiness, more so ever our target sensor zone exists in California where sunlight is of abundance hence our decision to use solar panels supplemented by fairly capacious power sources including Pb batteries. Environmental data sensors on the RISE platform include a temperature sensor LM61 from National Semiconductors and a CO_2 sensor (CARBOCAP GMT 220) from Vaisala.

Figure 2 shows the block diagram of the RISE sensor platform, hashed boundaries indicate the components that are currently under development. Table 1 lists the salient features of the RISE sensor platform.

We now discuss the individual components that make up the RISE platform in more detail.

3.1. The Chipcon CC1010 SoC

Measuring a midget 12 mm along side widths and just 1.2 mm high, the CC1010 is a feature packed SoC, supporting SPI and baseband decoding in hardware while also maintaining interoperability with existing platforms including the MICA.

The CC1010 is a true single-chip UHF transceiver with an integrated high performance 8051 microcontroller with 32 KB of Flash program memory. The CC1010 together with a few external passive components constitutes a powerful embedded system with wireless communication capabilities. The features of the Chipcon SoC [4] are as follows:

- High-Performance and low-power optimized 8051-core microcontroller that typically gives 2.5x the performance of a standard 8051. Idle and sleep modes for reduced power consumption. The system can wake up on interrupt or when ADC input exceeds a set threshold.
- A low current consumption fully integrated UHF RF Transceiver with programmable frequency and output power and low current consumption. It also has fast PLL setting for frequency hopping protocols, Manchester encoding and decoding in hardware and also RSSI output which can be sampled by the on chip ADC.
- 32 KB of nonvolatile Flash memory with programmable read and write locks for software security 2KB + 128 Byte of internal SRAM.
- Peripheral features include three channel, 10 bit ADC, programmable watchdog timer, real time clock with 32 KHz crystal oscillator, two programmable serial UARTs, master SPI interface, two counters and pulse width modulators, 26 configurable general-purpose I/O-pins and random bit generator and H/W DES encryption.

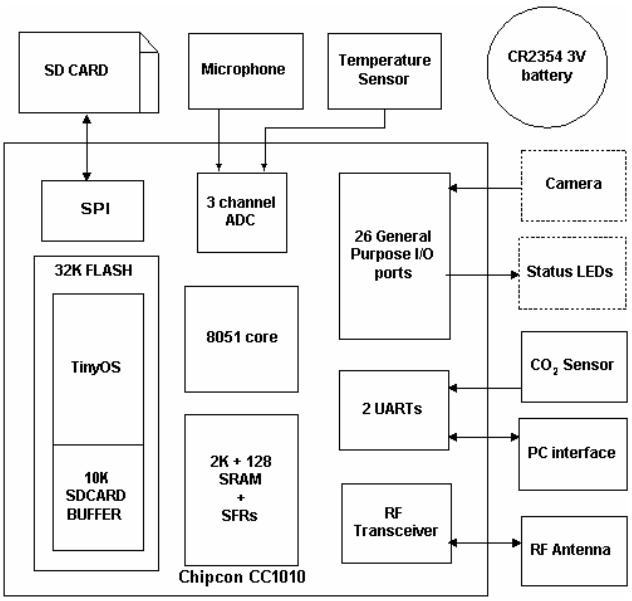

Fig2. The RISE platform block diagram.

3.2. TinyOS on RISE

TinyOS [8] provides an event based execution and a constrained runtime environment ideally suited to, and the present de-facto standard operating system for sensor network systems. The modular design of TinyOS is amenable to simple applications not requiring complex hardware resources.

The latest stable versions of TinyOS, tinyos-1.1, as also the NesC compiler, nesc v1.2alpha1, were ported on to the RISE platform. The starting point of the port was the Wisenet project [1], which had ported the older versions of the TinyOS and NesC. The newer versions of TinyOS and NesC now include support for clock synchronization, which is essential in indexing and storing the data on the flash.

Additionally, instead of using the *KeilμVision IDE* to manually generate the hex form to burn onto the chip, we decided to use the *SDCC* (small device C compiler), freely available in the public domain and integrated it such that the single make of the application creates a hex file ready to be burned on the on-chip flash. Fig 3 shows the flow diagram of the build process of an application for the RISE platform; boxes with hashed boundaries are the additional or modified components.

The C language file was produced by *nesc1.exe.* The script *nesc-compile* was modified to pass source code to the custom post-processor for RISE, *sdccppp.* This script extracts the 8051 specific parameters that were passed through the nesc compiler

Component	*Capability*
The MCU	
Processor	24 MHz 8051 core
On-chip flash	32 KB
Current Consumptions (Active, Idle, Power Down) at 14.7456 MHz	14.8 mA, 8.2 mA, 0.2 µA
The Radio	
On-chip Radio	300-1000MHz low power RF transceiver
RF transmission rate	76.8 kbits/s
Range	Upto 250m at 868/915 MHz
Current Consumptions for RF transceiver (Receive, Transmit at 10dBm)	11.9 mA, 26.6 mA
Typical SD-Card Specifications	
Current Consumption (Write, Read, Sleep)	80 mA, 60mA, 500µA
Access Time	200µsec (max)
Read or Write on 256-512 MB card	10MB/sec for both
The SPI bus	
Datarate	Programmable upto 3 MHz
The data block length	512 bytes

Table1. The RISE platform features

and invokes the *sdcc* compiler and the *packihx* tools with the relevant flags to generate the hex file that can then be stored on the flash program memory.

3.3. Interfacing the SD-Card

SD-Cards are commercial non-volatile flash memory storage devices about the size of a postage stamp (24mm by 32mm by 2.1mm). Their slim, compact design makes them an ideal removable storage solution for a wide range of devices like digital cameras, PDAs, and cellular phones These cards have built-in memory controllers and consume a low power while reading or writing and a miniscule amount when idle. Various other standards for flash memory exist, such as the Compact Flash (CF), XD Card, etc. CF cards communicate using a parallel bus, unsuitable for simple microcontrollers, while the XD-Card is devoid of a memory controller. SD-Cards on the other hand support the popular SPI (Serial Peripheral Interface) bus interface, which it inherited from the earlier generation Multimedia Cards. SPI is a serial data interface for a microcontroller which enables connecting peripheral devices with low programming effort. Additionally the very low cost of SD-Cards, approximately 6-10 cents per megabyte, make them a truly low cost solution.

Our choice of the SD-interface results in a simple board design and avoids soldering additional flash memory chips on the board. The connection from the platform to the SD-Card involves dedicating four I/O pins (Clock, Data In, Data Out, Chip Select) from the microcontroller to the SD-Card and three other power pins. Since all RISE platforms incorporate this memory interface, addition of auxiliary memory is as simple as inserting a SD-Card into the socket.

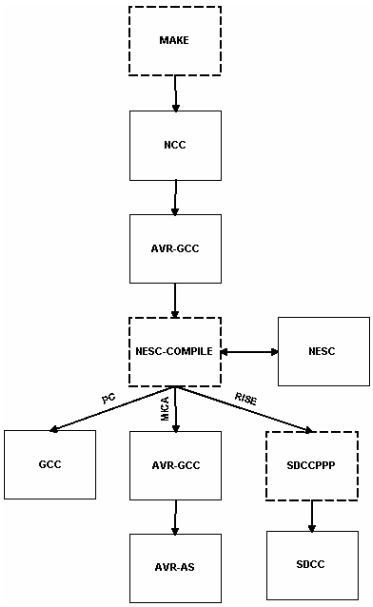

Fig3. The TinyOS flow diagram for a typical build

The microcontroller transfers data using the SPI protocol by linking to a wrapper component that provides read / write macros to facilitate data transfer to and from the SD-Card. The wrapper component in turn uses the on-chip SPI interface, to communicate with the SD-Card. Each write transaction to the card involves writing a 512 byte block of data, while reads may be arbitrarily sized up to a maximum of 512 bytes, at a maximum rate of 3Mbps. Because a small amount of sensor data would be generated per wake cycle, it would not possibly fill up the 512 byte block, and writing it thus would entail zero padding, and wastage of energy. Write efficiency is enhanced by first buffering the sensor data to the on-chip flash. A full buffer initiates a data flush from the on-chip flash to the SD-Card. Currently we allocate 10 KB of the on-chip flash memory for buffering while the rest is used for user data and program. We are currently working on developing tiny access method structures which allows for efficient sorted and random access to local data.

4. THE PARADIGM OF SENSE AND STORE.

4.1 Rationale behind Sense and Store

Typically the sensor network applications generate vast amounts of temporal data over very long time intervals that need to be collected and processed in a power efficient manner. Of the vast amounts of data generated, applications are typically interested in only aggregates, hence database approaches have been advocated [9]. We discuss how employing large memories on the sensor nodes can fulfill the requirements of these applications, in an overall methodology we call “sense and store”.

During normal conditions, the sensory data remains predictable with gradual changes, and hence is not of particular importance. But often the question that we are interested in answering is of the form *“When did we have the last 5 highest temperature readings?”* or *“What was the average temperature last week?”,* or consider our present scenario wherein sensors

are deployed in a forest to track climate conditions, and there is a forest fire. In this case the data set of interest is the one, say up to a month leading to the event. Hence, in most cases the queries request only a part of the sensed data, sorted by either the index (the timestamp in our case) or the value and the data of interest in just a fraction of the sensed data. Typically these sets involve temporal and top-k class of queries.

We aim to exploit this nature of sensory data in our methodology of sense and store and show that it is much more efficient to sense and store data on a large flash memory and transmit it only when requested. Thus instead of wasting large amounts of energy transmitting unnecessary data, we employ a large memory to store all of the data and do small local processing like update of current minimum, maximum or average values and bookkeeping of various sorted lists and indexes. It has been noted that communication costs a great lot when compared to local processing, as also the reduction in traffic simplifies network design significantly, thus permitting scalability to a large number of nodes. As we see in Section 6, the energy required for the transmitting one byte is roughly equivalent to executing 688 CPU instructions, and the cost of writing to the flash is less than 10% of the energy required to transmit the same amount of data, making local storage and processing highly desirable.

Efficient evaluation of top-k queries in our platform can be achieved by estimating some threshold below which tuples do not need to be processed or transmitted from the sensor nodes. The key idea is to transmit only the necessary information towards the querying node and to perform calculation in the network rather than in a centralized way. In this scheme, each sensor node maintains locally in the flash memory a window of m measurements. This window evolves with time, and therefore, once the limit of the available memory is reached, current aggregates should be transmitted to the sink, otherwise at each new time moment the oldest measurement is deleted. We note however that given the capacities of SD-Cards m can be very large. Registered queries can perform some local aggregation, if the correctness of the query outcome is not violated, before values are propagated towards the parent.

4.2 Providing Local Access Methods

Efficiently evaluating the queries described above requires efficient access to the data that is stored on the "external" flash memory. Therefore we plan to deploy certain access methods (indexes) directly at the sensor nodes. These access methods will serve as primitive operations for the efficient execution of a wide spectrum of queries. Since the flash card on each node v_i can only hold m pages ($o_{i0}...o_{im}$) the available memory is organized as a circular array, in which the newest o_{ij} pair always replaces the oldest o_{ij} pair. Note that this *sorted page organization* allows each sensor to have random or sorted access by timestamp in $O(1)$, without the requirement of any index.

Next we address how indexes become useful, when we need to have access by value as well as the challenges that need to be solved.

i) Random Access By Value: An example of such operation is to locally load the records that have a temperature of, say, 70F. In order to fetch records by their value we will use a static hash index.

ii) Sorted Access By Value: An example of such operation is to locally load the records that have a temperature between 50F-70F. In order to fetch records by their value we will use a simple B+ tree index. This index is a minimalistic version of its counterpart found in a real database system. It consists of a small number of non-leaf index pages which provide pointers to the leaf pages.

Depending on the sample rate of a query, we expect to have a number of insertion and deletion that need to be handled efficiently. For this purpose we plan to keep a buffer for a small set of leaf pages(each page is 512 bytes) in the MCU flash memory (which is 32KB).

4.3 Efficient Top-k Query Evaluation in RISE

We now sketch an algorithm to compute top-k queries which is designed to take full advantage of the capabilities of the sense and store paradigm. This algorithm, the *Threshold Join Algorithm*, decreases the number of values transmitted from each sensor, by using an additional probing and filtering phase.

More specifically, the algorithm consists of three phases:
1) the *Lower Bound* phase, in which the sink collects the union of the top-k results from all nodes in the network (denoted as $L_{sink}=\{l_1,l_2,...,l_o\}$, $o \geq k$),
2) the *Hierarchical Joining* phase, in which each node uses L_{sink} for eliminating anything that has a value below the least ranked item in L_{sink},
3) the *Clean-Up* phase, in which the actual top-k results are identified.

The use of local structures that index the data stored in each sensor significantly improve the execution of this algorithm, and can have a fundamental impact of the efficiency of the execution of other complex queries.

To validate the efficiency and applicability of our approach of evaluating top-k queries in our Sense and Store sensor framework, we have tested our algorithm in a Peer-to-Peer network collected at 32 sites in Washington and Oregon [2]. Each peer (node) maintained the average temperature on an hourly basis for 208 days between June 2003 and June 2004 (i.e. 4990 moments), and our query was to find the 10 moments at which the average temperature was the highest. We compared our approach with the Sense and Send (SS) approach (sending all data over the network), and our preliminary results indicate that the SS approach consumes an order of magnitude more network bytes than our Sense and Store approach when our top-k algorithm is applied. We also performed a comparison with a technique that transmits all data from the sensors but employs in-network aggregation in a hierarchical fashion to compute the average temperature of different time instances as the data are transmitted in the network. Our preliminary results show that our approach outperforms this technique by a factor of 5, in terms of network bytes consumed.

5. ENERGY EVALUATION

We calculate the energy gains that can be achieved by use of the sense and store methodology and contrast and compare storing a bit of data to sending it over the network. For a single hop, energy consumed by the radio to transmit a single bit is 0.094 nAh, and to receive is 0.042 nAh. On the other hand energy

required to write to the SD-Card memory is 0.007 nAh and to read is a mere 0.006 nAh.

Note that although the SD-Card current consumption is more than that of the radio (best case), the considerably higher data transfer rate (3 Mb/s as compared to 76.8 Kb/s) implies that the read or write operation on the flash occurs for a much shorter time and hence the energy expense for flash operations is less than 10% of the radio operations. If the fraction of data that turns out to be useful is denoted by x, and needs to be transmitted over the radio, then total energy needed to write all the data to the flash and read and transmit the relevant data can be expressed as

$$E=0.007 + x(0.094+ 0.006)\ nAh/bit$$

Hence % energy savings can be expressed as

$$S\%= 92.14 – 105.89x$$

As an example assume that the fraction of the useful data is 50% of the data sensed. In that case energy savings is of the order of 39%. The break even point is when the data requested is around 87% of the total sensed data. Any application that uses less than 87% sensed data could immediately start saving energy if it starts to save the data on the flash card.

Also, at a speed of 14.756 MHz and with an instruction of 4 clock cycles, the processor is able to execute 688 instructions in the same amount of energy required to transfer a byte.

The savings become even more pronounced when we consider the effect on the overall network. In the sense and send approach each node not only sends unnecessary packets but also acts as relays for other messages which have to be carried hop by hop to the sink, this phenomenon effectively floods the network with unnecessary packets. Less traffic would imply less contention and interference which in turn translates into additional energy savings and improved lifetime of the network.

6. IMPLICATIONS

Distributed, wireless, microsensor networks will enable myriad applications for sensing the physical world. Considering past efforts at experiments similar to the one discussed in Section 2, we adjudge that using the sense and store methodology will be very helpful in most of the data acquisition applications in the sensor domain.

One of the first successful experiments involving environmental and habitat monitoring using sensor nodes was conducted by University of California, Berkeley at the Great Duck Island, Maine [11]. Various sensor motes fitted with light, temperature, barometric and acoustic sensors were utilized for the task. The sense and send paradigm although largely successful, resulted in a lifetime of almost seven months for the experiment.

Our sense and store paradigm can be implemented in such situations to advantage, and to increase the longevity of similar experiments by virtue of saving power and cutting down on communication costs, as most sensor data in such specific experiments is a slowly varying, redundant time series[11] [13]. Moreover if the type of queries are similar to the k-maximum / minimum, average, etc such queries could be most effectively handled by sense and store as the nodes may continuously calculate and store these values in their local memories.

Exemplifying the data received at a typical daylight sensor, that for most parts of the day, data across the light sensor increases gradually until midday and starts decreasing slowly thereafter, and remains zero at night [13]. Hence this data may be tightly compressed and stored during the day, while also calculating certain statistically important values such as maximum, minimum, average, etc, that would be transmitted to the base station at night, effectively reducing communication traffic during the day. Similar variations can be expected for various other geo-climatic variations such as the temperature and pressure at a location. Similarly, the measurement of event longevity, such as infrared monitoring of animal dwellings [11] [13] involves long intervals of near constant data, followed by the end of the event, and is largely a binary phenomenon, i.e. either the event persists or has lapsed. Sense and send may not be justifiable, due to the requirement of communicating the data throughout the lifetime of the event, and there seldom lies any information for the duration of the event. On the other hand sense and store can locally store and process the complete details of the event while communicating only relevant details for its reconstruction (such as start, stop time).

Data fidelity and power consumption are two opposing criteria for sensor systems and most energy constrained sensor networks tend to balance between data transmission and storage. Many data such as photographic, acoustic, etc need to be downsampled before being transmitted over the wireless channel, thus sacrificing the quality of the sensed data, possibly hindering future data mining. Local storage at the sensors thus helps to preserve the original data, which can then be manually retrieved and correlated at the end of the experiment.

Another problem afflicting sensor networks is the inability of modeling the actual behavior and lifetime of the nodes in the field, particularly due to lack of original operational data / post mortem data [13]. Due to the availability of large local storage, strategic vital statistics and diagnostic data vis-à-vis the motes may be stored for post mortem analysis thus aiding our understanding of actual deployment.

Tiered sensor networks have been proposed for various environmental monitoring deployments such as the habitat sensing array for biocomplexity mapping, as proposed for James Reserve [6]. The platforms in the tiered network constituting the highest level of the hierarchy are relatively high performance computing devices communicating with the next in hierarchy, the tags (sensor nodes similar in capabilities to RISE). Sense and store enable the tags to locally store and manage data, as broadcasted from minute memory-less sensors (devoid of any receive circuitry), which just sense and send, owing to their utterly simplistic and constrained designs.

Overall the benefits of sense and store are perceptible in sensor networks deployed for monitoring time series data with predictable queries and where post mortem analysis and collection of data would be undertaken.

Keeping these requirements in mind, we are currently calculating and storing the running average, daily min-max, daily average, in the on-chip flash, which are then stored into the SD-Card past elapse of twenty-four hours (RISE incorporates timekeeping, thanks to the realtime clock on the platform[4]). Currently under implementation are the top-k and range queries which would be completed after due field runs to determine a practicable value of the query window, along with generating diagnostics information (battery voltage) from the ADC.

7. RELATED WORK

In the development of distributed sensor networks a large variety of prototype systems have been implemented and tested. The systems are spread all over the cost, power, functionality and form factor space. Table 2 contrasts the RISE platform to some of the low cost, low power contemporary sensor platforms. In [7] a survey of current platforms and a classification according to hierarchies is presented. In the low cost, low power generic sensor spectrum we notice the absence of a platform which boasts a large memory. Some other platforms in the high end, high bandwidth category feature high end processors and expandable memories, some including support for flash. However these high end platforms are more of a tradeoff between the requirements of the traditional low power embedded sensors, which can be deployed in large numbers, and the substantial requirements of high end applications and networking, as also the convenience, usability of a desktop development environment.

Systems which propose a declarative approach for querying sensor networks include TinyDB [10] and Cougar [15]. These systems achieve energy reduction by pushing aggregation and selections in the network rather than processing everything at the sink. Both approaches are optimized for sensor nodes with limited storage and relatively short-epochs, while our techniques are designated for sensors with larger memories and longer epochs. In Data Centric Storage (DCS) [14] data with the same name (e.g. humidity measurements) are stored at the same node in the network, offering therefore efficient location and retrieval. However the overhead of relocating the data in the network can become expensive if the network generates GBs of data.

8. CONCLUSION AND FUTURE WORK

In this paper, we present RISE, a networked sensor system featuring a large storage memory, thus paving the way for a new paradigm in power conservation for applications that collect data over long periods of time viz. environmental and habitat monitoring. Since adequate memory and power conservation is often a significant design criterion for developing a sensor platform, we believe that adoption of sense and store along with efficient query mechanisms provide significant energy benefits to a wide spectrum of applications that can use the large memory for local storage, aggregation and compression before transmitting the results towards the sink. Moreover, the stored diagnostics data, retrieved from sensors would help us accurately determine causes of failure and aid development of better and efficient sensing platforms. Presently these options are available only in the sensor systems belonging to the class of power hungry and complex platforms. In addition we expect that with the provision of efficient access methods, our new framework would enable wireless sensor systems to cope with highly demanding new generation applications that have not yet been addressed adequately. In future we plan to investigate the effectiveness and limitations of our framework on a wide spectrum of sensor platforms.

Platform	MCU	MCU, Active Current	On Chip RAM	On Chip Flash	Aux. Mem.
RISE	24 MHz 8051 core	14.8 mA	2KB	32KB	Upto 1 GB
RENE	4 MHz AT90s8535	6.4 mA	512B	8KB	-
MICA MICA 2DOT	16 MHz ATMega 128L	8 mA	4KB	128KB	512K B
Telos	8 MHz TI MSP430	0.56 mA	2KB	60KB	512K B
EcoNode	16 MHz 8051 core NRF24E1	3 mA	4K	32KB EEPR OM	-
iBadge	4 MHz ATmega 103L	5.5 mA	4K	128K	Option -al 64 KB

Table2. Contemporary generic sensor platforms

9. REFERENCES

[1] http://cegt201.bradley.edu/projects/proj2003/wisenet/
[2] http://www-k12.atmos.washington.edu/k12/grayskies/
[3] http://www.ccb.ucr.edu/
[4] http://www.chipcon.com/
[5] J. R. Agre, L. P. Clare, G. J. Pottie, and N. Romanov, “Development Platform for Self-Organizing Wireless Sensor Networks,” *Proceedings of SPIE’s 13th AeroSense, Unattended Ground Sensor Technologies and Applications Conference,* April 1999.
[6] A. Cerpa, J. Elson, D.Estrin, L. Girod, M. Hamilton, and J. Zhao, “Habitat monitoring: Application driver for wireless communications technology”, *ACM SIGCOMM Workshop on Data Communications in Latin America and the Caribbean*, 2001.
[7] J. Hill, M. Horton, R. Kling and L. Krishnamurthy, "The platforms enabling wireless sensor networks", *Communications of the ACM*, 47(6):41–46, 2004.
[8] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. E. Culler, and K. S. J. Pister, “System architecture directions for networked sensors”, In *Proceedings of ASPLOS*, pages 93–104, Boston, MA, USA, Nov. 2000.
[9] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, ”TAG: A Tiny AGgregation service for ad-hoc sensor networks”, *Fifth Symposium on OSDI ’02*, Boston, Dec. 2002.
[10] S. Madden, M.J. Franklin, J.M. Hellerstein, and W. Hong, “The design of an acquisitional query processor for sensor networks”, *ACM SIGMOD*, 2003, San Diego, CA, 2003.
[11] A. Mainwaring, J. Polastre, R. Szewczyk, and D. Culler, “Wireless sensor networks for habitat monitoring”, *ACM Workshop on Sensor Networks and Applications*, 2002.
[12] R. Pinilla, N. Navarro, and M. Gil, “TinyOS: A Case of Study for Adaptable Embedded Applications Based on Sensor Networks”, *Technical Report UPC-DAC-2004-6, Technical University of Catalonia*, February 2004.
[13] J. R. Polastre , “Design and Implementation of Wireless Sensor Networks for Habitat Monitoring”, *MS thesis*, 2003 available at: http://www.cs.berkeley.edu/~polastre/papers/masters.pdf
[14] S. Shenker, S. Ratnasamy, B. Karp, R. Govindan, D. Estrin, "Data-centric storage in sensornets", *ACM SIGCOMM Computer Communication Review*, vol 33-1, 2003.
[15] Y. Yao, J. E. Gehrke, "Query Processing in Sensor Networks", *CIDR'03*, Asilomar, CA, 2003.