

Dartmouth College

Dartmouth Digital Commons

Dartmouth College Undergraduate Theses

Theses and Dissertations

6-1-2003

Power Conservation in the Network Stack of Wireless Sensors

Michael De Rosa
Dartmouth College

Follow this and additional works at: https://digitalcommons.dartmouth.edu/senior_theses



Part of the [Computer Sciences Commons](#)

Recommended Citation

De Rosa, Michael, "Power Conservation in the Network Stack of Wireless Sensors" (2003). *Dartmouth College Undergraduate Theses*. 32.

https://digitalcommons.dartmouth.edu/senior_theses/32

This Thesis (Undergraduate) is brought to you for free and open access by the Theses and Dissertations at Dartmouth Digital Commons. It has been accepted for inclusion in Dartmouth College Undergraduate Theses by an authorized administrator of Dartmouth Digital Commons. For more information, please contact dartmouthdigitalcommons@groups.dartmouth.edu.

Power Conservation in the Network Stack of Wireless Sensors

Michael De Rosa

Dartmouth Computer Science Technical Report TR2003-458

Abstract: Wireless sensor networks have recently become an incredibly active research area in the networking community. Much attention has been given to the construction of power-conserving protocols and techniques, as battery life is the one factor that prevents successful wide-scale deployment of such networks. These techniques concentrate on the optimization of network behavior, as the wireless transmission of data is the most expensive operation performed by a sensor node. Very little work has been published on the integration of such techniques, and their suitability to various application domains. This paper presents an exhaustive power consumption analysis of network stacks constructed with common algorithms, to determine the interactions between such algorithms and the suitability of the resulting network stack for various applications.

Keywords: wireless sensor networks, energy efficient design, network protocols

Table of Contents

Introduction	1
Hardware	1
Software	2
Power Consumption	3
Operating Modes	4
Application Layer	5
Network Layer	6
MAC Layer	7
Data Link Layer	7
Experimental Configuration	9
Notation	9
Metrics	10
Results/Analysis	11
Discussion	13
Next Steps	17
Conclusions	17
Acknowledgements	18
Bibliography	19
Appendix A: Tables	21
Appendix B: Figures	24

Introduction

Wireless sensor networks have recently generated much interest in the computer science research community. The development and deployment of these networks poses unique challenges in the fields of networking, distributed computing, and hardware design. Much has been made of the fact that wireless sensor nodes operate under severe power constraints [1], and many novel network algorithms [2,3] have been developed to minimize the power costs associated with the wireless transmission of data inside these networks. Specifically, many routing and MAC algorithms have been developed expressly for the purpose of power conservation. Unfortunately, an efficient algorithm at one level of a network stack may not guarantee an efficient network stack. There is thus a need to analyze the behavior of network algorithms in the context of a complete network stack. Such system-level analysis is not well represented in current publications, although there are notable exceptions [4]. This paper presents the results of such an analysis, comparing different network stacks constructed from common network algorithms.

Hardware

Experiments were targeted to the U.C. Berkeley MICA device [5], which has become one of the standard research platforms for wireless sensor research. The Berkeley device is relatively mature, and is readily available. MICA is constructed around an Atmel ATMega128L microcontroller, a popular 8-bit MCU [6]. The

ATMega128L's features are summarized in **Table 1**. The radio interface for the device is an RFM TX1000 transceiver module. This device operates in the 916Mhz ISM band, and provides 40 Ksymbol/sec throughput with an ideal range of 100 meters [7]. Variable range attenuation is provided by a software-addressable digital potentiometer. The TX1000 utilizes an antenna fabricated on the surface of the MICA circuit board. MICA's interface to mission-specific sensors is provided via a high-density connector on the main circuit board, and sensors are available for light, sound, temperature, acceleration, and magnetic fields. Custom sensors also may be deployed, either via the ATMega128L's 10-bit A/D converters or its I2C interface. MICA also provides 3 LEDs, a voltage boost converter, a 4 Mbit serial Flash RAM, and a coprocessor for over-the-air reprogramming. The device is powered by 2 AA alkaline cells, which have a rated charge capacity of 2850 mAh [8], although more than 50% of this is at a voltage below 2.4 V.

Software

As a complement to the MICA hardware, U.C. Berkeley developed the TinyOS platform [9,10]. TinyOS is a component-based OS specifically designed for wireless sensors. The component-based nature of the OS makes it easy to replace portions of the network stack, or even the entire stack, without disrupting other elements of the system. The TinyOS communications architecture uses variable length packets, with a maximum payload length of 32 bytes. The packet headers, automatically included by TinyOS, include an address field, a type field, a 2-byte CRC, a transmission strength pulse, and an automatic acknowledgement mechanism [11]. TinyOS supports compilation from the

same code to multiple hardware platforms, as well as to the included Nido simulator. This simulator provides basic event and radio-propagation models. To improve the verisimilitude of the simulation, we developed custom sensor and radio modules. The sensor model provides pseudo-realistic photosensor and temperature sensor data, while the radio component adds the capacity for random bit errors (overhearing and collisions being already supported by the base simulator). Additionally, the logging capabilities of the simulator were modified to permit capture of all RF mode changes.

Power Consumption

In a wireless sensor network, the limiting variable is almost always power. All deployable wireless sensor nodes currently use some form of battery technology. When dealing with single devices, the replacement of batteries is not a major issue. In wireless sensor systems comprising hundreds or thousands of nodes, replacing the batteries of any particular node may be impractical or expensive. Such is especially the case when the sensor nodes are integrated into the superstructure of a building, or deployed in a battlefield's fire corridor. Extensive effort has been devoted to the creation of efficient power-scavenging systems for use in wireless sensor networks, via solar power or thermal differentials, but to date such research has not produced hardware that is suitable for field use. Given that every action taken by a wireless sensor node exhausts some of its power reserves (or power budget, in the case of energy-scavenging nodes), it is imperative to limit how often energy-intensive tasks are performed. The two components of the MICA system that draw the most power are the processor and the RF transceiver,

whose power consumption characteristics are summarized in **Tables 2 and 3**. As the differential in power consumption between sleep and active is several orders of magnitude, it follows that reducing the duty cycle of the CPU and RF transceiver would directly impact the field lifespan of a wireless sensor node. If all of the components of a MICA node are in power-save mode, its theoretical maximum lifespan is 2917.7 days. This decreases to 23.75 days if the CPU is running at 100% duty cycle, and 6.9–13.5 days if the CPU and RF transceiver are both active continuously.

Operating Modes

To test the response of various communications algorithms, two application scenarios were constructed. These scenarios are not intended to be exact implementations for a particular task, but rather serve as exemplars for different usage modes.

The first application scenario is that of passive environmental monitoring. A sensor network is deployed to provide wide-area, continuous monitoring of some environmental variable for a period of time. For the purposes of this research, the sampled variable was temperature, which was sampled every 100ms at each node. In a 100-node network, ten 10-bit samples/second creates a total information flow of 10,000 bits/second (exclusive of packet framing and encoding overhead), which is well under the 40Ksymbols/s channel capacity. All nodes report their data back to the base station, where further post-processing and data uploading is performed.

The second application scenario is that of fire detection and reporting. In this scenario, nodes monitor their sensor every second. If a node senses a value above a certain threshold, it signals to its 1 and 2-hop neighbors. All activated nodes send their current light readings to the base station, then resume normal function. For this experiment, the sensor used was a photodetector, which approximates a smoke or carbon monoxide detector. Network traffic in this experiment is more sporadic and less stream-oriented than that of the first application scenario.

Application Layer

In the passive monitoring application scenario, application-level payload compression was implemented. Three well-known algorithms [12,13] (plus no compression) were selected as candidates. Distributed and wireless-sensor-specific compression techniques were also considered [14]. The first algorithm selected was Huffman encoding, with a dictionary built specifically for the expected sensor input. This provided a compression ration of approximately 2:1. In the second algorithm, the first-order residual of the data was taken, and Huffman encoding was applied to the residual. This compression was also performed using a custom dictionary, and the results were similar to those of the first algorithm. In the third algorithm, CDFM with an adaptive delta function was used. In this lossy compression algorithm, each 10-bit sensor reading is reduced to a 1-bit value, reflecting whether the value is larger or smaller than the previous one. Errors due to compression loss averaged 0.178 bits/sample.

Network Layer

Two representative multi-hop routing algorithms were chosen as candidate algorithms. These algorithms were chosen to represent the two major types of multihop routing algorithms: beacon- and table-driven [15]. The first was DSDV, a beacon-driven algorithm [16]. In DSDV, the base station broadcasts beacons at regular intervals, which are propagated through the network by smart flooding. As each node receives the beacon, it updates how many hops away it is from the base station, and for which node it received the beacon. To transmit data to the base station, a node sends that data to the node that forwarded the most recent beacon. Similar steps are performed at each node, leading back to the base station.

The second candidate algorithm was DSR, an active route-acquisition algorithm [17]. DSR is actually an ad-hoc routing algorithm, but with slight simplifications so that multihop communication was only supported between a node and the base station. In DSR, a node that does not have a route, but wishes to send data to the base station sends a route request packet. This packet is flooded, and as it is forwarded from node to node, a record is kept on the packet of the nodes it has visited. This prevents looping and continual flooding. Once the request reaches the base station, the order of the recorded hops is reversed, and a route reply is unicast along the chain of addresses leading to the originating node. As each node receives a route reply, it can use the node from which it heard the reply as the first hop when attempting to communicate with the base station.

MAC Layer

Three passive MAC algorithms were selected as MCA layer candidates. The first algorithm was carrier sense with random backoff, the most basic MAC technique. The second algorithm was identical to the first, with the addition of a random wait period before the transmission attempt. Values for the random wait windows were taken from the original TinyOS networking components. These values were 30-61 bit intervals for random backoff, and 80-127 bit intervals for random initial wait. The third MAC algorithm was an implementation of the MAC1p protocol, designed for low-power wireless devices [18]. This protocol implements adaptive rate control by probabilistically dropping packets at transmit time. Upon a successful transmission, the probability of a packet being dropped is decreased, while an unsuccessful transmission increases the probability. Routing control packets are not dropped. Random wait and backoff are also implemented in this protocol. Values for random intervals and probabilities were taken from TinyOS components, and suggestions presented in [18].

Data Link Layer

Two different sets of candidate algorithms are implemented at the data link layer: automatic retransmission requests (ARQ) and forward error correction (FEC). These are two complementary techniques for maintaining link quality. As previously mentioned, the TinyOS network components support automatic acknowledgement of correctly

received packets. If a transmitted packet is not correctly received at any node, ARQ will attempt to retransmit the packet up to three times.

In addition to ARQ, forward error correction provides an added layer of reliability. FEC attempts to detect and correct limited numbers of errors introduced by RF channel noise and interference [19]. Four encoding protocols were selected as candidate algorithms. The simplest is no encoding whatsoever. Bits to be transmitted are passed directly to the RF hardware. This provides no error detection or correction capability, but incurs no overhead. Unfortunately, the choice to use no encoding is not feasible in practice. The RFM transceiver recommends that signals be DC-balanced, in order to improve the receiver's signal-to-noise ratio (SNR.) Thus, the simplest practical protocol is Manchester encoding. This encoding scheme DC balances the output signal, at the cost of halving the effective bandwidth. Manchester encoding also provides no error correction capabilities, although it can detect any single-bit error. A popular encoding scheme that provides some error recovery facilities is SEC-DED. SEC-DED is a block code that provides DC balancing, single error correction, and double error detection. SEC-DED reduces effective bandwidth by $2/3$. Finally, a code known as EVENODD was implemented. EVENODD is an erasure-channel based error detection and recovery scheme originally developed for RAID arrays [20,21,22]. Data encoded with EVENODD is subsequently encoded with Manchester encoding, to create an erasure channel that can detect all single bit, and many 2-bit errors. EVENODD can correct up to two erasures, which corresponds to two non-consecutive bit errors in the encoded byte. EVENODD also provides DC balancing (via Manchester encoding). EVENODD reduces effective bandwidth by $2/3$.

Experimental Configuration

Four sets of experiments were carried out. In each, a 100-node network was simulated for 30 seconds. All possible combinations of encoding, MAC, ARQ, routing and (in the case of passive monitoring) compression were tested. Each application scenario was tested with both a grid and a random topology. The grid topology consisted of a 10x10 rectangular grid, with no “wrapping” at the edges. The random topology was a unipartite graph, with each node having 4 connections to random neighbors. The random topology was held constant throughout the tests, as was the governing random seed for the simulator. The configuration of each experiment is summarized in **Table 4**. These experimental parameters were chosen to reflect a typical small-scale distribution of sensor in both planned and randomly distributed configurations. During simulation, data was captured for each node on the following variables: length of time in each RFM mode, number of bits corrupted by channel static, number of corrupt packet receptions, number of retransmissions, number of attempted transmissions, and number of transmissions successfully received at the base station.

Notation

As there are 192 candidate network stacks that can be created from the space of candidate algorithms, a simple notation has been adopted to express them concisely. The notation is detailed fully in **Table 5**. Briefly, the notation for a specific network stack consists of a 5-tuple of numeric values, expressed as <C-R-A-M-E>, where C is the

compression algorithm used (or X if the experiment used the fire detection scenario), R is the routing algorithm selected, A indicates the presence or absence of ARQ, M is the MAC algorithm, and E is the encoding scheme (FEC). Mean values from aggregate data slices are represented using the star operator (*). For example, the subset of stacks that use ARQ and Manchester encoding is represented as <*-*-1-*-1>.

Metrics

Two main metrics were used to evaluate the performance of the candidate network stacks. The first of these is Jain's metric for multi-hop network fairness [23], which is expressed as

$$F = \frac{(\sum X_i)^2}{n \sum (X_i)^2} \quad (1)$$

where X_i is the ratio of actual receptions at the base station to expected receptions at the base station, and n is the number of nodes sending to the base station. In this metric, 1.0 represents perfect fairness, while 0.0 represents the domination of the entire channel by one node. The second major metric used is power cost / event. In the case of the passive monitoring experiments, this cost is mAs/kbit of information, while in the fire detection scenario, it is mAs/alert received at the base station. In this metric, lower power cost is desired.

Results / Analysis—Individual Experiments

Data from each of the four experiments was collected and analyzed according to the metrics described earlier. A summary of findings within each experiment is found in **Tables 6 and 7**. It is interesting to note that all of the stacks that are optimal for either metric in a particular experiment are part of the subset $\langle *-0-0-0-0 \rangle$, indicating that the simplest components at each layer of the network stack provide the best performance. Unfortunately, the choice of no encoding at the radio level, while providing the best results in simulation, is not a practical decision for field deployment. We therefore remove all stacks of the form $\langle *-*-*-*0 \rangle$ from consideration, and again find the optimal stacks for each experiment, with regard to both power consumption and reception fairness. These results are summarized in **Table 8**. Here we see that in 7 of 8 cases, Manchester encoding, the simplest practical encoding, provides the best performance. In half of the cases, carrier sense with random backoff provides the best performance, while in the other half, the optimal stack uses carrier sense plus initial wait time. Surprisingly, in only one case does the addition of ARQ create an optimal stack. In the network layer, DSDV appears to provide superior performance to DSR. Finally, Huffman residual compression provides the most reception fairness, while CDFM provides the lowest power cost per kilobit of data.

Results/Analysis—Multiple Experiments

Looking at the aggregate data for all of the experiments, we can analyze the data in several ways. By taking subsets of the data, one can isolate the effect that various algorithms have on both power consumption and reception fairness. These results are shown in **Figures 1 and 2**. In evaluating the effect of algorithms on network fairness, we see that the algorithm that gives the best performance varies with the application scenario. In the passive monitoring scenario these algorithms are (in descending order through the stack): DSR routing, ARQ active, carrier sense + wait MAC, and Manchester encoding. It is interesting to note that, with the exception of Manchester encoding, none of these algorithms appears in the stacks that had the highest fairness in either Experiment 1 or 2. In the fire detection scenario, the fairest algorithms appear to be: DSDV routing, ARQ inactive, carrier sense MAC, and no encoding, which corresponds to the fairest stack for Experiments 3 and 4.

This inter-application variance is not the case with power consumption, where the simplest algorithm appears to always have the best performance. Note that in **Figure 2**, many of the error bars are a significant fraction of the recorded value. This indicates that the choice of a particular algorithm is not a strong indicator of performance.

Confining ourselves to the passive monitoring scenario, we can compare the performance of the different compression techniques in the two topologies studied. In both Experiment 1 (random topology) and Experiment 2 (grid topology), CDFM provides the lowest power cost, while Huffman residual compression provides the highest reception frequency [**Figures 3 and 4**].

To obtain a quantitative measure of how the optimal stacks from each experiment perform against the “default” stack suggested by the authors of TinyOS, we compare the two optimal stacks for each experiment against the best-performing member of the subset $\langle \ast-\ast-0-2-2 \rangle$, which represents the default stack specified in TinyOS. We extend this comparison to include the “best compromise”, the stack that has the lowest average ranking in both power consumption and reception fairness for a given experiment. The results of these comparisons are shown in **Figures 5 and 6 and Tables 10 and 9**, and the list of candidate stacks used for these comparisons appears in **Table 8**.

Discussion / Conclusions

The measurements and comparisons of the previous section can be used to derive certain conclusions about the behavior of wireless sensor networks in various application scenarios, and to begin to form a set of guidelines for the selection of a network stack for a given task.

Beginning with the application layer, we note that the algorithm that provides the most compression (CDFM) provides both the lowest power consumption and the lowest network fairness. Why is this? At 1 bit/sample, each packet can hold 184 samples. During the course of a 30-second experiment, each node transmits only one packet. Thus, network usage is extremely low (as each packet only transmits once), but if any packet is lost to corruption or interference or local congestion, then no readings are received at all from the originating node. This is especially apparent when CDFM is combined with DSR routing. DSR routing does not establish a route until the first

transmission request. Thus, when a node has filled its packet with 184 samples and requests transmission, DSR will send a route request packet. As the simulator only slightly randomizes the start time for each node, many of these route request packets will be transmitted within several seconds of each other. Such transient collision will probably destroy most of the route requests, preventing the return of route replies and the delivery of nodes' sensor readings. An algorithm that appears to provide a reasonable compromise between power consumption and reception fairness is Huffman residual encoding. This algorithm can fill a packet with 9-35 samples per packet, and this range introduces significant randomness into the timing of transmission requests. Additionally, as all of a node's readings are now not contained in one packet, the fairness of the system is not as adversely affected by the loss of one packet as it is under CDFM.

At the network layer the two choices are DSDV, a beacon-driven routing algorithm, and DSR, an active route acquisition algorithm. DSDV appears to consume less power than DSR in most cases, although this may be due more to the experimental conditions than to the performance of the algorithm itself. DSR floods the network whenever it receives a transmission request and does not have a route (due to an error or having been recently initialized). In a large network with a high data volume, these additional packets merely contribute to the congestion of the network. Additionally, as 75% loss rates are not uncommon in sensor networks, the likelihood of a route request reaching the base station and subsequently returning intact is at times rather low. This should not be taken as a categorical failure of DSR, however. DSDV has a lower initial cost, as only one node is flooding, but since the base station periodically floods, the cost of routing is not limited to initialization and error correction, as in DSDV. In an

initialized network, with low error rates, it is therefore conceivable that DSR could outperform DSDV. Further research into the parameters necessary for this crossover is suggested.

At the LLC layer, we first analyze the performance of ARQ. ARQ is designed to ensure delivery in cases where a transient error condition has resulted in the corruption of a sent packet. This condition is usually a bit error, or the random collision of two packets. In high-traffic scenarios, corruption of a packet appears to usually be the result of congestion. In this case, retransmission will only increase local congestion and power consumption. This can be seen in the increased power consumption associated with the activation of ARQ. The effect of ARQ on reception fairness is more interesting. In the passive monitoring scenario ARQ increases reception fairness, while in the fire detection scenario, the reverse is true. This appears to be the result of the two different application scenarios' bandwidth usage. In the passive monitoring application, bandwidth usage is constant and high. In this environment, retransmission can increase the chance that a particular packet will be successfully received, which is especially important for distant nodes, whose transmissions must traverse multiple hops to reach the base station. In the fire detection scenario, bandwidth usage is characterized by intermittent and simultaneous transmissions by a cluster of nodes. In this scenario, retransmission is likely to block transmission attempts by a neighbor, who may only attempt to transmit once or twice during the course of the experiment.

Moving on to MAC algorithms, we see very interesting results when comparing the power consumption of the three algorithms. The simplest algorithm, carrier sense with random backoff, has much lower power consumption than either of the more

complex algorithms. More complex algorithms create a lower effective bandwidth. In carrier sense + random initial wait, the transmission of every packet is delayed, even if the channel is free at that moment. Similarly, in MAClp, the transmission of any packet may be randomly cancelled by the adaptive rate control mechanism. The simplest algorithm, carrier sense alone, is a purely opportunistic user of bandwidth. If immediate transmission is possible, this algorithm is the only candidate that will take advantage of it. This seems to indicate that a greedy algorithm can best exploit transient dips in bandwidth usage. By removing the overhead associated with cooperation between nodes, the total channel capacity is effectively increased.

At the encoding layer, one can examine the tradeoff between packet size and error recovery capability. In a low-traffic environment dominated by single-bit errors, a complex encoding scheme that provides multiple-bit error recovery provides the best performance, as the increased packet length will not create more collisions. In an environment where congestion, and thus collision, is the most common source of error, FEC (in the form of SEC-DED or EVENODD) is of little use. As collisions produce multiple consecutive errors, FEC cannot correct them. Additionally, as error-correcting codes impose a 50% length increase over Manchester encoding alone, the chance of collisions is increased while the effective bandwidth is decreased. This would indicate that, in congested environments, the benefits that FEC provides are not worth the additional cost in bandwidth.

Next Steps

This series of experiments explored the behavior of several network algorithms in different simulated application scenarios. To improve the reliability of the results, the experiments should ideally be repeated with different topologies, random seeds, numbers of nodes, and simulation durations. By simulating under these varying conditions, one could decrease the possibility that the patterns of behavior indicated in the data are the result of a particular configuration, and not of the experimental scenario itself. As the experiments are currently implemented, they measure the startup cost of a network in addition to its operational costs. In a deployed network, the startup costs are of much less concern than the steady state power consumption of the system. To properly measure the steady state performance of the system, it would be necessary to modify the simulator to begin with an initialized network. Finally, the experiment could be improved by the addition of more network algorithms and application scenarios, and the optimization of the existing algorithms. This is especially true for the EVENODD encoding scheme, which appears to suffer from an error in decoding.

Conclusions

By testing the performance of many network stacks under two application scenarios, we see that the most effective stacks appear to be those constructed from the simplest algorithms. Using simple components, one not only reduces power consumption, but also computation, another highly limited resource in this class of systems. In systems

where the aggregate performance of the network is more important than the performance of any single node, algorithms designed to ensure reliable communication may actually decrease the performance of the system as a whole.

These findings have significant implications for the development and deployment of wireless sensor networks, which have traditionally been designed using assumptions taken from ad-hoc and cellular wireless systems. As these new sensor networks are more power- and computation-limited than their predecessors, priorities in their design are quite different than those of traditional wireless networks.

Acknowledgements

My sincerest thanks to my advisor Bob Gray, and to the members of my thesis committee: Bob Gray, Daniela Rus, and David Kotz. Also to the U.S. Airforce ActComm project, for providing funding to purchase hardware. Thanks to Professor Daniela Rus, Professor David Kotz, Qun Li, and Ronald Peterson for technical advice. Thanks to my family, my friends, and my brothers at ΣN for all their support, and for putting up with the long hours and endless requests for proofreading. And finally, thanks to Jenn, without whom none of this would have been possible.

Bibliography:

- [1]K. Kalpakis, K. Dasgupta, and P. Namjoshi. "Maximum Lifetime Data Gathering and Aggregation in Wireless Sensor Networks," In Proceedings of IEEE Networks'02 Conference, 2002.
- [2]C. E. Jones, K. M. Sivalingam, P. Agrawal, and J.-C. Chen, "A survey of energy efficient network protocols for wireless networks , " Wireless Networks, vol. 7, no. 4, pp. 343--358, 2001.
- [3]Q. Li, J. Aslam, and D. Rus, "Online power-aware routing in wireless ad-hoc networks," in Proceedings of the Seventh Annual International Conference on Mobile Computing and Networking, July 2001, pp. 97-- 107.
- [4]D. Ganesan, B. Krishnamachari, A. Woo, D. Culler, D. Estrin, and S. Wicker. "Complex behavior at scale: An experimental study of lowpower wireless sensor networks," Technical Report UCLA/CSD-TR 02-0013, UCLA Computer Science, Jul, 2002.
- [5]J. Hill and D. Culler, "A wireless embedded sensor architecture for system-level optimization," <http://today.cs.berkeley.edu/tos/papers/MICA ASPLOS NAMED.pdf>
- [6]Atmel, Inc, <http://www.atmel.com/>, "ATMega128L Data Sheet-Complete"
- [7]R.F. Monolithics, Inc. <http://www.rfm.com/>, "ASH Transceiver TR1000 Data Sheet"
- [8] Duracell, Inc., <http://www.duracell.com/oem/Primary/Alkaline/mx1500.asp>, "Duracell MX1500 Product Information"
- [9]J. Hill "A Software Architecture Supporting Networked Sensors," M.S. Thesis. University of California at Berkeley, 2000.
- [10]J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. "System architecture directions for networked sensors," In Proceedings of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS IX), pages 93--104, Cambridge, MA, Nov, 2000.
- [11]P. Buonadonna et. al. "Active Message Communication for Tiny Networked Sensors," <http://today.cs.berkeley.edu/tos/papers/ammote.pdf>
- [12]R.M. Gray "Quantization," IEEE Transactions on Information Theory, Vol. 44 No. 6, October 1998
- [13]D.A. Lelewer and D.S. Hirschberg, "Data compression," ACM Computing Surveys 19,3 (Sept.): 261-266, 1987.

- [14]L. Doherty “Algorithms for Position and Data Recovery in Wireless Sensor Networks,” M.S. Thesis, University of California at Berkeley
- [15]S. Ramanathan, and M. Steenstrup, “A Survey of Routing Techniques for Mobile Communication Networks,” *Mobile Networks and Applications*, vol. 1, No. 2, pp. 89 - 104, 1996.
- [16]C.E. Perkins and P. Bhagwat, “Highly Dynamic Destination-Sequenced Distance Vector Routing (DSDV) for Mobile Computers,” *ACM SIGCOMM: Computer Communications Review*, vol.24, no.4, pp.234-244, October 1994
- [17]D.B. Johnson and D.A. Maltz, “Dynamic Source Routing in Ad Hoc Wireless Networks,” in *Mobile Computing*, edited by T. Imielinski and H. Korth, chapter 5, pp.153-181, Kluwer Academic Publishers, 1996
- [18]A. Woo and D. Culler, “A Transmission Control Scheme for Media Access in Sensor Networks ,” In *ACM/IEEE International Conference on Mobile Computing and Networks (MobiCOM) 2001*.
- [19]D.J. Costello Jr. et. al. “Applications of Error Control Coding,” *IEEE Transactions on Information Theory*, Vol. 44 No. 6, October 1998
- [20]M. Blaum et al.: “EVENODD: an efficient scheme for tolerating double disk failures in RAID architectures,” *IEEE Transactions on computers*, Vol. 44, No 2, pp. 192-201, February 1995.
- [21]D. Feng et. al. “Improved EVENODD Code,” *Proceedings of 1997 IEEE International Symposium on Information Theory*, IEEE Computer Society (Cat. No.97CH36074), June 29-July 4, 1997, Ulm, Germany, p.261
- [22]P.J. Havinga: “Energy efficiency of error correction on wireless systems,” *proceedings IEEE Wireless Communications and Networking Conference (WCNC'99)*, September 1999.
- [23]R. Jain, “Fairness: How to measure quantitatively?,” *Tech. Rep. 94-0881, ATM Forum*, Sept. 1994.

Appendix A: Tables

Table 1: Capabilities of the ATmega128L

Function	Value
Clock Speed (103 Emulation Mode)	4Mhz
Registers	32x8 bits
Program Space	128K
RAM	4K
Counters	2 8-bit, 2 16-bit
ADC	10 8-bit
USARTs	2
SPI	1

Table 2: Electrical Characteristics of the ATmega128L

Mode	Power Supply Current (max)
Active	5 mA
Idle	2 mA
Power Down	40 μ A

Table 3: Electrical Characteristics of the RFM TX1000

Mode	Power Supply Current (typical)
Receive	3.8 mA
Transmit	12 mA
Sleep	0.7 μ A

Table 4: Experimental Parameters

Experiment	Scenario	Topology
1	Passive Monitoring	Random
2	Passive Monitoring	Grid
3	Fire Detection	Random
4	Fire Detection	Grid

Table 5: Network Stack Notation

<Compression-Routing-ARQ-MAC-Encoding>

Compression Value	Meaning
0	No Compression
1	Huffman Compression
2	Huffman Residual Compression
3	CDFM Compression
X	No Compression (Fire Detection)
*	Wildcard

Routing Value	Meaning
0	DSDV Routing
1	DSR Routing
*	Wildcard

ARQ Value	Meaning
0	ARQ inactive
1	ARQ ACtive
*	Wildcard

MAC Value	Meaning
0	Carrier Sense
1	Carrier Sense + Initial Wait
2	MAClp
*	Wildcard

Encoding Value	Meaning
0	No Encoding
1	Manchester Encoding
2	SEC-DED Encoding
3	EVENODD Encoding
*	Wildcard

Table 6: Reception Fairness of Simulated Network Stacks

Experiment	Min Fairness	Mean Fairness	Max Fairness	Opt. Stack
1	0	0.153419896	0.750815457	<2-0-0-0-0>
2	0	0.178453721	0.67581228	<2-0-0-0-0>
3	0.01010101	0.16820018	0.402194569	<X-0-0-0-0>
4	0.018123569	0.177758279	0.381730453	<X-0-0-0-0>

Table 7: Power Consumption of Simulated Network Stacks (mAs/data volume)

Experiment	Min Power C.	Mean Power C.	Max Power C.	Opt. Stack
1	22.47491018	545.8167887	12752.36798	<3-0-0-0-0>
2	23.66605634	424.0640327	2112.599257	<3-0-0-0-0>
3	11.44092044	421.9059404	7728.643523	<X-0-0-0-0>
4	27.38309018	134.1653885	495.6902725	<X-0-0-0-0>

Table 8: Optimal Stacks (No Encoding Removed)

Experiment	Opt. Fairness Stack	Opt. Power Stack	Best Compromise
1	<2-0-0-0-1>	<3-0-0-1-1>	<2-0-0-0-1>
2	<2-0-0-1-1>	<3-0-1-1-1>	<1-0-0-1-1>
3	<X-0-0-1-1>	<X-0-0-0-1>	<X-0-0-1-1>
4	<X-0-0-0-3>	<X-0-0-0-1>	<X-0-0-2-1>

Table 9: Reception Fairness Comparison of Optimal and Standard Stacks

Experiment	Opt. Fairness	Opt. Power	Best Compromise	Standard
1	0.60529818	0.12121212	0.60529818	0.3663133
2	0.55157515	0.1038961	0.52427795	0.37943791
3	0.36988167	0.30529172	0.36988167	0.26926889
4	0.33833177	0.27555794	0.33270464	0.22105952

Table 10: Power Consumption Comparison of Optimal and Standard Stacks

Experiment	Opt. Power	Opt. Fairness	Best Compromise	Standard
1	31.5897374	79.6325983	79.6325983	93.0211466
2	37.025219	138.905469	119.68214	179.265868
3	26.3870554	28.178825	28.178825	55.2800598
4	53.6101403	84.8613952	57.9058671	109.289486

Appendix B: Figures





