Power-Efficient Adaptable Wireless Sensor Networks

John Lach[1], David Evans[2], Jon McCune[3], Jason Brandon[1], Lingxuan Hu[2]

University of Virginia Departments of Electrical and Computer Engineering[1] and Computer Science[2], Charlottesville, VA 22904

Carnegie Mellon Department of Electrical and Computer Engineering[3], Pittsburgh, PA 15213

Abstract

Wireless sensor networks represent a new data collection paradigm in which adaptability plays an important role. Typical sensor network scenarios involve scattering a large number of wireless nodes from an aircraft across an area of interest. The nodes then form a network through which collected data is routed to a base station. Adaptation is necessary to deal with the unpredictable network topologies that result from sensor node scatters and to manage resources (energy in particular) efficiently in response to changing conditions and requirements. The hardware flexibility required for dynamic adaptation is traditionally achieved with software-based processors or field programmable gate arrays (FPGAs), both of which come with significant energy, area and performance costs when compared to application-specific integrated circuits (ASICs). We therefore introduce a small-scale reconfigurable design technique that minimizes these costs by efficiently integrating small amounts of application-specific reconfigurable logic within primarily fixed-logic circuitry. This technique provides the flexibility necessary for the adaptations required of wireless sensor networks without the penalties associated with processors and FPGAs. This paper makes the case for small-scale reconfigurability by investigating several different types of adaptation in wireless sensor network applications that allow applications to deal with unpredictable network topologies and tradeoff between network longevity and fidelity, security and latency.

I. INTRODUCTION

In typical wireless sensor network applications, a large number of sensor and actuator devices (nodes) are scattered by an aircraft across a target area. A high power base station is then used by an operator to control the behavior of and receive data from the nodes via a wireless network. Such applications pose a number of design challenges for sensor networks:

- Nodes are commonly deployed from an aircraft in a rapid, ad hoc manner, preventing designers from making assumptions about where particular nodes will be located, what the network topology will be, and the role of each node in the topology.
- Application requirements and environmental conditions often change during network operation.
- Nodes often fail during network operation, due to depleted energy, destruction, or movement out of transmission range.
- Given that the nodes are wireless and ideally quite small, they have very limited energy, most of which is consumed by data transmission.
- The large number of nodes in sensor networks, their limited lifetime, and their inherent disposability require a low per unit cost.

In this paper, we explore how these challenges can be addressed through dynamic network adaptability and mechanisms for achieving certain types of adaptation at both the application and node levels. Section II describes the lifetime of a wireless sensor network and considers how requirements differ throughout that lifetime. Section III considers how adaptability can be used to deal with unpredictable network topologies, adjust for changing application requirements, recover from node failures, and improve energy management. In Section IV, we consider approaches to node design that provide the necessary flexibility for adaptability with low unit cost and minimum energy consumption. This requires the integration of hardware design techniques that provide high efficiency (in terms of cost, energy, and performance) but minimal flexibility with techniques providing the flexibility required for node adaptation but limited efficiency.

II. WIRELESS SENSOR NETWORK LIFETIME

Before we discuss the *types* of adaptations that can be performed in wireless sensor networks, we first explore the various *stages* at which adaptations and design for adaptability can be performed. We can divide the lifetime of a sensor network into several stages, during which application requirements and environmental conditions may change or become fixed. At different stages, designers will have the option to give up flexibility to reduce cost and improve performance. Although the network lifetime is more accurately continuous, we can loosely identify the following distinct phases:

Device Design Time. A single node design may be used for many different network applications. The communications and data processing aspects of a node are likely to be similar even if the particular application changes. By designing a single node that can be used for many applications, the overall design and manufacturing costs will be reduced through amortization and economies of scale. At this stage, designers must predict the range of applications and attempt to design a single device that serves all of those applications adequately. To reduce costs and power and improve performance, flexibility can be sacrificed, but that will limit the possible applications. Companies including Crossbow Technology and Dust, Inc. are beginning to sell generic devices for wireless sensor networks.

Application Design Time. After the particular desired application for a network is established, much of the device flexibility is no longer required, as the application constrains the expected data processing and communication requirements. For example, an application might be tracking moving objects in a region [1] or monitoring the temperature and humidity of a habitat [2]. The tracking application emphasizes low latency; the habitat monitoring prioritizes longevity. For certain applications where the scale is huge or the performance requirements are severe, it may be cost-effective to design a network node specifically for the application. For most applications, though, this would be too expensive. Instead, the application will be designed around available generic devices that can be cheaply configured to particular design parameters.

Scenario Design Time. The scenario of a wireless sensor network application determines the number of devices and the environment in which the devices will be operating. An example scenario might be tracking vehicles in a particular parking lot or monitoring the habitat in a certain bird migration zone. A given scenario constrains the likely communication and data processing requirements even more than an application does and may establish particular latency, security, fidelity and longevity requirements.

Deployment. For many applications, the actual and relative locations of devices will not be determined until the network is deployed. Based on those locations, devices may change their behavior. For example, devices in high-density regions may enter sleep mode and wait for their neighbors to lose power. Other devices may adapt to act as message forwarding nodes instead of (or as well as) processing or sensing nodes.

Operation. Once the network has been deployed and is operating, the behavior of individual devices may need to adapt to either changing requirements or conditions. A weather monitoring network operator may decide high-fidelity temperature readings are more important than humidity readings and send control messages into the network to use communication and data processing energy accordingly. Network devices may sense a possible intruder and change communication modes to use a higher level of encryption. Nodes may exhaust their energy or may otherwise malfunction requiring topology adaptation.

A designer of a sensor network application must consider what degree of adaptability is necessary at various stages in the application's lifetime. The more flexibility that is available, the more freedom there will be to adapt the behavior and properties of the nodes and network. That freedom is not without cost, however. Designs that provide unused flexibility waste resources, primarily energy and cost. These issues are explored in the following two sections.

III. THE CASE FOR ADAPTATION

In many sensor network applications, the requirements are not known precisely enough at deployment time to make tradeoffs required for maximum network utility. Even during operation, requirements will change in response to varying conditions. The challenge, then, lies in designing sensor network applications that can dynamically adapt to changing requirements by selecting different points in the design space for trading off resource consumption for fidelity, security and latency. This is inconsistent with the traditional view of algorithms in computer science that focuses on their functional properties (inputs and outputs) and characterizes their performance based on the time or space the algorithm requires for a given input size. For wireless sensor networks, it is important that we have algorithms whose behavior can be characterized in more precise ways and be parameterized in ways that change the properties of the algorithm. Since sensor networks are deployed by scattering devices over an area, it is important that nodes are able to adapt their behavior to their topological role in the network. As nodes fail or move, they will have to alter their behavior accordingly. Since nodes in a wireless sensor network are untethered and typically must run on small batteries, a primary factor in determining the utility of a network is how well it manages energy consumption. As nodes fail due to energy depletion, a network can quickly become useless. Therefore, it is often desirable to adapt energy consumption in ways that increase network longevity by sacrificing other application properties.

Next, we describe examples showing the importance of adapting to network topology, and illustrate how applications can reduce energy consumption (and increase application longevity) dynamically by adapting fidelity, security and latency parameters according to changing application requirements.

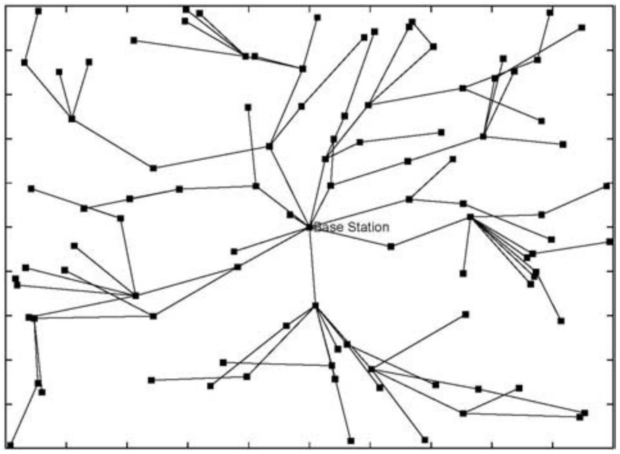
A. Network Topology

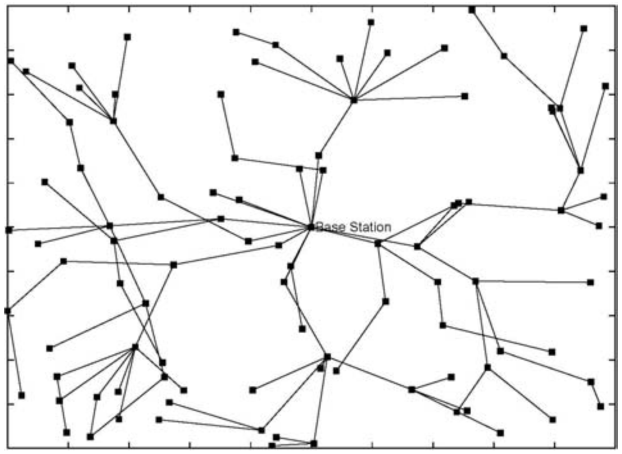
The topology of a sensor network is normally unknown until deployment. For example, sensor nodes may be dropped out of an airplane over a large area. Designers of sensor network applications cannot assume they know where particular nodes will land, and often manufacturing and deployment costs will require that all of the devices are initially identical.

Once the network is deployed, however, the location of the nodes is known (and in some cases, the devices will not move again once they are deployed). Based on happenstance, some nodes will land close to the base station and others will land further away; some geographical areas will have a high density of nodes, and others may have no nodes at all; some nodes will have many neighbors within communication range, others may have only a few.

In order for information from the far away nodes to reach the base station, the nodes need to form a multi-hop routing tree through connecting nodes to the base station. A simple way to form a routing tree is for the base station to transmit a short-range message identifying itself. All nodes that receive that message are level 1 nodes: their parent is the base station itself. In the next step, each level 1 node transmits a message that identifies itself as a level 1 node. Upon hearing this message, a node that is not already a level 1 node, will become a level 2 node and select the sending level 1 node as its parent. (To balance the routing tree, if a node that is already a level 2 node receives an announcement message from a level 1 node that is not its parent, the level 2 node will arbitrarily select one of the parents based on randomly assigned node identifiers.) This process continues until every node reachable from the base station has a parent node. To avoid transmission collisions, time segments are divided according to level and node identities.

Figures 1a and 1b show two examples of routing trees that form following this protocol starting from 100 randomly scattered nodes. To extend the useful lifetime of a sensor network, nodes must adapt to their role in the routing tree, which cannot be predicted before deployment or even just based on location. A node with many children will not survive long if it forwards all of its children's messages instantly and completely. A node with many neighbors may be able to coordinate with those neighbors to avoid redundant messaging and take turns running in sleep mode to save energy. Some of these behavior changes are application-specific and may need to be carefully designed for a particular network application.





Figures 1a and 1b: Initial routing trees formed from random deployments of 100 nodes.

A simple strategy for increasing the longevity of a network is to have the network routing adapt as nodes fail. When a node's parent fails, the child node sends out an orphaned message to find a new parent node to continue sending data towards the base station. Any functioning node that hears the orphaned message responds with its level, and the orphaned node selects the most suitable parent. Figures 2 and 3 illustrate the impact of parent adaptation on network utility. Note that in Figure 2 many live nodes are unable to communicate with the base station, whereas with parent adaptation as shown in Figure 3 even though several aggregation point nodes have died, their children have found alternate parent nodes to maintain a path to the base station. Some live nodes are temporarily disconnected, but will soon find new parents in response to orphaned messages.

Our simulations illustrate that adaptive routing techniques can maintain network connectivity even as nodes fail. To further improve network longevity, other adaptations may trade off application properties to conserve energy.

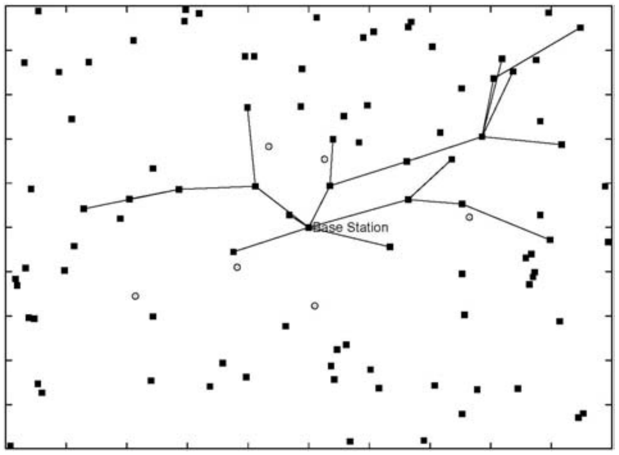


Figure 2: Network from Figure 1a after transmitting 300 requests without adaptation. Nodes that have failed due to energy exhaustion are shown as circles.

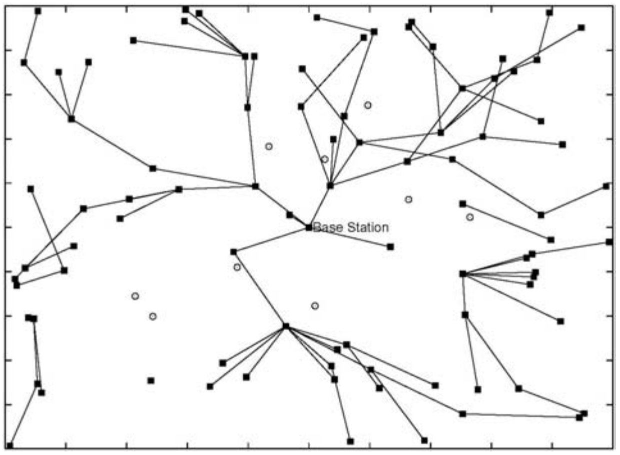


Figure 3: Network from Figure 1a after transmitting 300 requests with orphaned child routing adaptation.

B. Fidelity

For a simple example of the need for dynamic adaptation for energy management, consider a network that monitors an environment with cameras. Each node in the network takes pictures of its surrounding area, compresses each image using JPEG compression, and transmits the data to its parent in the network. While each leaf node only transmits its own images, a parent node must also transmit the images of all of its descendents. A high-level node with many children, grandchildren and great-grandchildren is responsible for transmitting a large amount of data. The nodes closest to the base station (and highest in the routing tree), will have the most data to transmit and will quickly exhaust their available energy. After the highest-level nodes have been depleted, useful information can no longer reach the base station.

The JPEG compression algorithm can be parameterized to control the block size and tradeoff image quality for execution time and (more importantly) the size of the output data and the amount of energy required to transmit it. At the network topology level, the routing tree could be transformed to reduce the data forwarding pressure on nodes with a large number of descendents. Parents could use image processing techniques to aggregate images from their children into a single image, or they could use filtering to selectively forward only interesting images. At the node level, the image compression can be parameterized for different image qualities and compression ratios to trade off energy consumption and image fidelity. Nodes can remain in a low power mode (highest compression, lowest image quality) until the base station commands a group of nodes to switch to a higher resolution.

Figure 4 shows how adapting JPEG block size affects the useful lifetime of such a network. JPEG 1 provides the highest compression and lowest image quality; JPEG 8 provides the least compression and best image quality. The network starts with 400 nodes that are all able to transmit their images to the base station through the routing tree. As nodes run out of energy, they are no longer able to transmit their own or their descendents' images. When a parent's energy is depleted, the routing tree automatically adapts to have its children find different parents to whom to send their images, but a live parent node is not always reachable. Given that data transmission dominates this application's energy consumption, the smaller transmissions required for JPEG 1 images allow the network to keep a larger number of nodes over time. Using JPEG 1, the base station receives at least 100 images for each of the first 1400 requests, but by the 300th request using JPEG 8, fewer than 100 images are received. An adaptable network could switch between JPEG 1 and JPEG 8 based on available energy or commands from the base station.

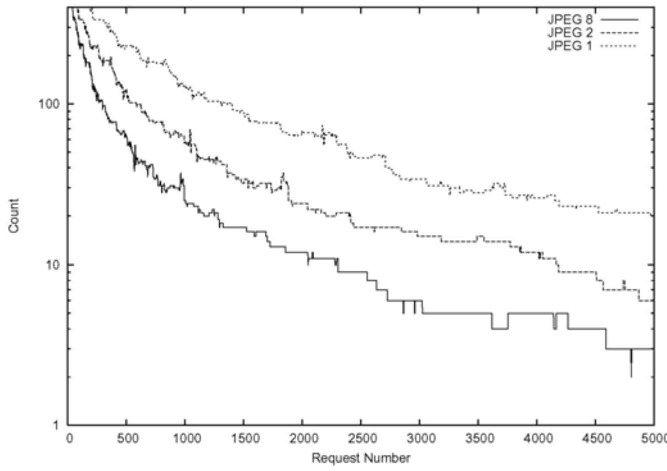


Figure 4: Network fidelity for different image compression ratios. The vertical axis shows the number of nodes whose image reaches the base station in response to each request. Initially, there are 400 nodes, scattered randomly throughout an area. Results are the average of 12 simulated executions. Our simulations are built using extensions to GloMoSim [3] to simulate a wireless sensor network. Normalizing the compressed transmitted image size at 1 for JPEG 1, JPEG 2 and JPEG 8 were of size 2.34 and 4.68, respectively [4].

As this example shows, we can sacrifice application fidelity to enable a network to provide meaningful readings for a longer time period. In other situations, it may be more important to provide high assurance that readings are correct over a short period of time.

C. Security

Most modern symmetric ciphers (including AES) can be parameterized to control the number of encryption rounds. Increasing the number of rounds strengthens the security of the cipher, but requires more computation (and hence, more time and energy). Figure 5 shows the effect of adjusting the number of encryption rounds on the longevity of a wireless sensor network application. Each transmission must be encrypted, but the number of rounds is made variable to tradeoff security and processing energy, just as various JPEG compression ratios may be used for different power modes. An adaptive application could use different encryption strengths (with their own keys) to adapt the costs associated with encryption to the security requirements of the transmitted data.

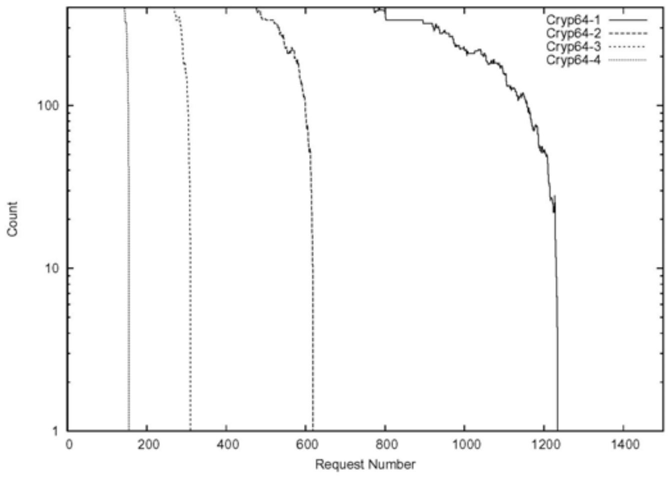


Figure 5: Effect of encryption strength on sensor network longevity. Cryp64-n represents encryption with 2^{n+1} rounds (e.g., Cryp64-4 is 32 rounds). The vertical axis again shows the number of nodes whose data reaches the base station through the routing network after a certain number of requests.

D. Aggregation

To reduce the energy consumed forwarding messages, sensor networks can perform calculations inside the network to determine aggregate results and forward those instead. Aggregation collects results from several nodes and calculates a smaller message that summarizes the important information from a group of nodes. For example, suppose the operator is interested in the average of some value in the network. An inefficient way to find this would be for every node to send its reading to the base station (often over multiple forwarding hops), and for the base station to calculate the average of all readings received. A more efficient way to collect the same information would be for intermediate nodes to forward the calculated average value of the readings they receive along with a count of the number of readings it incorporates. Several recent research efforts have explored different aggregation protocols for sensor networks assuming a trusted environment including directed diffusion [5], LEACH [6], greedy aggregation [7], Cougar [8] and TAG, an in-network aggregation service for TinyOS motes that supports an SQL-like language for expressing aggregation queries over streaming sensor data [9].

In general, aggregation protocols allow wireless sensor network applications to reduce energy consumption in exchange for increased latency and possibly lower fidelity. To aggregate responses, inner nodes must wait until several readings have been received before they are able to calculate an aggregate result and forward it to the next node. In some cases, the aggregation function may lose information (for example, sending averages instead of forwarding all readings means the base station does not receive enough information to construct a histogram).

Aggregation also makes authentication more difficult. Aggregating nodes must be able to combine multiple results into a single value, but a single compromised node should not be able to disrupt readings from an entire subtree. We have developed a technique (illustrated in Figure 6) that delays aggregation one hop to provide some of the energy-saving benefits of aggregation without sacrificing the ability for the base station to authenticate readings [10]. Keys are revealed after each time step is completed (and then invalidated so they may not be reused), so intermediate nodes can verify message authentication codes used during the previous step. As shown, a conspiracy of two consecutive nodes is required to forge readings of other nodes. Delaying the aggregation additional hops increases the communication costs but makes the protocol resilient to additional classes of attacks.

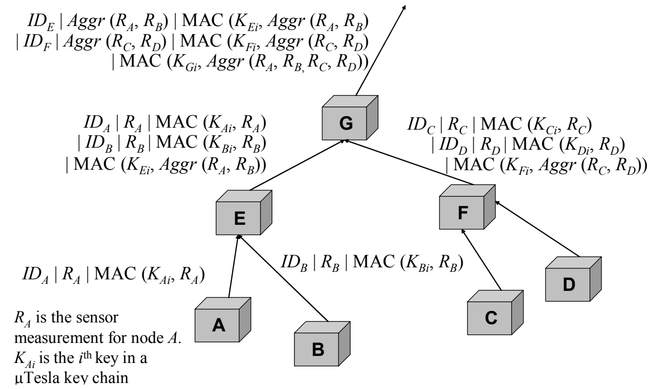

ID_E | $Aggr$ (R_A, R_B) | MAC (K_{Ei}, $Aggr$ (R_A, R_B)
| ID_F | $Aggr$ (R_C, R_D) | MAC (K_{Fi}, $Aggr$ (R_C, R_D)
| MAC (K_{Gi}, $Aggr$ (R_A, R_B, R_C, R_D))
G
ID_A | R_A | MAC (K_{Ai}, R_A)
| ID_B | R_B | MAC (K_{Bi}, R_B)
| MAC (K_{Ei}, $Aggr$ (R_A, R_B))
ID_C | R_C | MAC (K_{Ci}, R_C)
| ID_D | R_D | MAC (K_{Di}, R_D)
| MAC (K_{Fi}, $Aggr$ (R_C, R_D))
E
F
ID_A | R_A | MAC (K_{Ai}, R_A)
ID_B | R_B | MAC (K_{Bi}, R_B)
D
C
A
B
R_A is the sensor measurement for node A.
K_{Ai} is the ith key in a µTesla key chain

Figure 6: Data aggregation with authentication. Each leaf node forwards its data reading and a message authentication code (MAC) to its parent. Parents retransmit readings and MACs, along with a calculated aggregate value and MAC to grandparents. Grandparents can verify the aggregate value and need not retransmit the individual data readings and MACs.

Figure 7 illustrates the impact of authenticated data aggregation of network longevity. In this case, no time aggregation is used, so latency is not sacrificed. Aggregation functions may be specific to particular sensor network applications, but data aggregation can be generalized to many applications. A combination of the above techniques can be used to realize aggregation methods that tradeoff latency, security and fidelity for energy consumption.

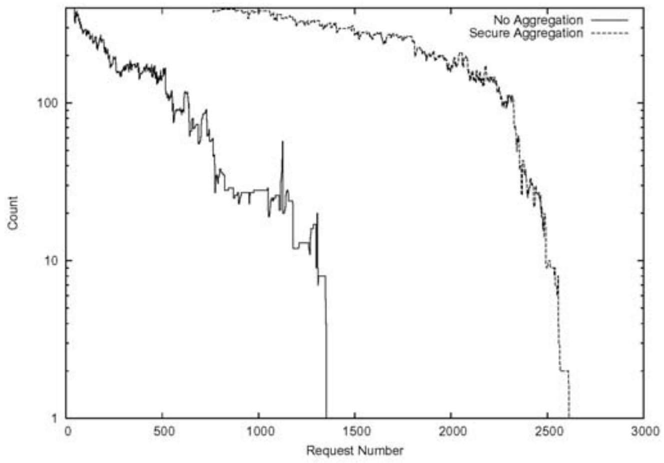


Figure 7: Effect of aggregation on network longevity.

IV. Designing Adaptable Nodes

Most wireless sensor networks are composed of nodes implemented with embedded processors that execute software-based instructions. For example, the MICA wireless sensor motes developed at UC Berkeley and marketed by Crossbow Technology each contain an Atmega 128L processor, which is a low-power microcontroller, running the TinyOS operating system [11]. These and other processor-based nodes are capable of functional adaptation simply by loading and executing various software programs. However, this virtually unlimited flexibility comes at a price, as software algorithms executed on processors typically consume more energy, are more expensive per unit for large volumes, and have lower performance than application-specific integrated circuit (ASIC) implementations of those algorithms. Given the limited energy, cost bounds and real-time requirements of many sensor nodes, ASIC-based nodes would provide significant benefits. However, fixed-logic ASICs do not have the flexibility to reap the benefits provided by dynamic adaptability illustrated in the previous section. Because of this efficiency/flexibility tradeoff inherent in node design, a wide range of implementation options are worth consideration.

A. Processing Hardware

The two efficiency extremes in implementing a single algorithm are direct hardware implementation using a fixed-logic ASIC and software instructions executed on a general-purpose processor (GPP). The custom nature of an ASIC makes it extremely efficient (in terms of energy, area, and performance) for executing a single algorithm, as it does not have any flexibility. GPPs allow for maximum flexibility and are therefore relatively inefficient per task. Since flexibility is expensive, finding the implementation that just satisfies the flexibility needs of an application will ensure highest efficiency. Considering that the adaptations a wireless sensor node must perform are limited, the excessive flexibility provided by the embedded processor in the MICA motes translates into reduced efficiency.

Modern hardware implementations include a number of design options between GPPs and ASICs, and the proper implementation technique must be chosen for providing the necessary adaptability with maximal efficiency. Given what information is known at different points in a node's lifetime, a hybrid device composed of several of these implementation types likely yields maximum efficiency. The following are distinct implementations of processing hardware, from the flexibility of GPPs to the efficiency of ASICs:

General-Purpose Processor (GPP). A processor that is designed for general use, such as those used in desktop computers. A GPP can perform any computable function (except as limited by available energy, memory and time) and provides a general instruction set that is not targeted to any particular application. The processor hardware is fixed, but the instruction memory can be loaded and re-loaded with various programs. GPPs can be high performance but require large chip area and are extremely power hungry.

Application-Specific Processor (ASP). A processor that is designed for a specific application. ASPs still execute software instructions, but the instruction set is designed for the execution of certain types of programs. They still are typically universal computers, but their performance on applications outside of their domain is reduced. Their efficiency performing the target applications is much higher than that for GPPs, but they still suffer the penalties inherent to software-based execution.

Digital Signal Processor (DSP). A type of ASP whose instruction set architecture and datapath are designed to handle the computationally intensive nature of digital signal processing. While the instructions can be changed to manipulate data in various ways, the strict instruction set and datapath cannot be easily used for non-DSP applications. DSPs have increased efficiency for related applications, but remain bounded by their execution of software instructions.

SRAM-Based Field Programmable Gate Array (FPGA). A structured array of reconfigurable logic and interconnect. Logic functions are implemented using SRAM-based lookup tables, and programmable interconnections are made through SRAM-gated pass transistors. FPGAs can be reconfigured a large number of times, including in the field, and can implement a wide range of algorithms in hardware. However, their general-purpose nature results in less dense logic, longer delays, and higher energy consumption than the following implementations.

Antifuse-Based FPGA. Similar to the SRAM-based FPGA, except these devices can only be programmed one time. Interconnections and logic functions are configured onto the device by blowing the appropriate set of fuses. While a non-programmed device can be configured to implement a wide range of algorithms, this flexibility is lost after it is configured. The anti-fuse nature of these devices make them much more efficient (in terms of energy, cost and performance) than SRAM-based FPGAs.

Application-Specific Integrated Circuit (ASIC). Fixed-logic hardware implementations of algorithms. They are designed to perform only a single task, but they perform it with the highest efficiency when well designed.

The coarse granularity of this set of distinct design options does not offer suitable choices for the efficiency/flexibility tradeoff that is central to wireless sensor node design. We therefore explore a new heterogeneous design technique that maximizes the amount of fixed-logic in a node (and therefore the efficiency) while providing enough flexibility for node and network adaptation. As opposed to the general-purpose reconfigurable fabric of FPGAs, heterogeneous small-scale reconfigurability (SSR) finely integrates small amounts of application-specific reconfigurable logic and interconnect with fixed-logic, providing only the flexibility that is required for the nodes to enable the necessary adaptations. As a result, SSR provides the necessary flexibility while achieving efficiency approaching that of ASICs. Hybrid FPGAs containing ASIC cores surrounded by general-purpose reconfigurable fabric arose from the same principle, but SSR provides more application-specific and fine-grained integration between fixed and reconfigurable logic for even greater efficiency.

The preservation of flexibility must be considered for the adaptations at different points in a node's lifetime, as discussed in Section II. Increasingly more information is known about a node and its role in a network at each successive stage in its lifetime. At the node's device design time, little may be known about its future network, application, or deployment. As a result, much of the node must be flexible. However, characteristics that are common to any wireless node can and should be implemented in fixed-logic. Once the network application and scenario are known, a significant portion of the previously flexible node can be permanently configured. Therefore, antifuse-based logic can be used in the node design and permanently configured once this additional pre-deployment information is known. As a result, this portion of the node can be made more efficient than would be possible with an SRAM-based reconfigurable logic implementation and significantly more efficient than software executed on a GPP.

Once the nodes are deployed, even more information is known. Based on whether or not this information will remain constant throughout the network's lifetime, the associated logic may also be permanently configured, thereby enabling again the use of antifuse-based logic. However, SRAM-based reconfigurable logic is required for portions of the node that may require multiple adaptations. Therefore, SRAM-based logic is required for operation adaptations, such as the different power modes in the JPEG compression example above, as further adaptations may always be required.

Understanding the various parts of a node's design that require adaptation at different points in its lifetime is the key to maximally efficient implementation. As much should be implemented with fixed-logic as possible; that which can be permanently configured once more information is available should be implemented using antifuse-based logic; portions of the node requiring multiple dynamic adaptations should be SRAM-based; and maximum programmability can be provided when necessary by limited support for executing software instructions.

B. Cost of Excessive Flexibility

Wireless sensor network application utility is primarily a function of battery life, making the efficient use of energy the primary consideration in node implementation, followed by performance for real-time considerations and area for sensor network applications,.

Figure 8 shows the utility of the image collection network described above with various node implementations. Specifically considered is the implementation of the JPEG compression algorithm. Efficiency is necessary for energy, area, and performance, but flexibility is required for dynamic parameter adaptation. Although this particular application is dominated by transmission energy consumption, it is clear the energy required for the JPEG compression processing has an effect on the network fidelity, as the less energy-efficient node implementations (GPP and FPGA) result in reduced fidelity. The SSR implementation shown is based on a reconfigurable architecture for adaptive wireless image communication [4]. Of course, the ASIC provides the longest utility, but it is not capable of switching JPEG parameters for fidelity adaptation.

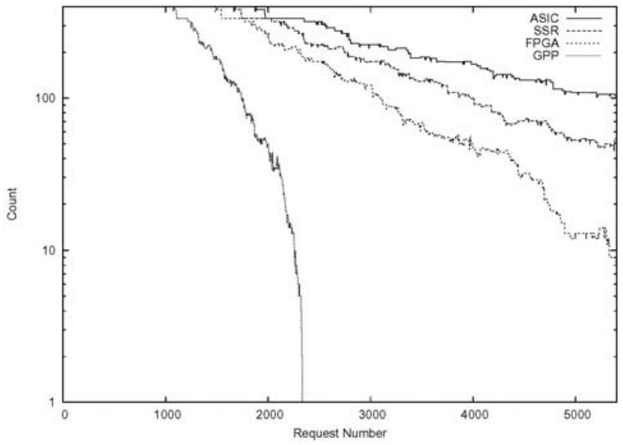


Figure 8: Impact of excessive flexibility on image collection network lifetime. As in Figure 4, the vertical axis is the number of nodes whose image reaches the base station after a request. The processing energy consumption of SSR nodes is normalized to 1, and the ASIC, FPGA, and GPP node processing energy consumptions are 0.01, 2, and 6, respectively [4]. Each node maintains a constant compression ratio and performs image aggregation at a ratio of $\sqrt{n}$, where *n* is the number of images at each node.

The advantages of more efficient node implementations are even clearer for applications that require a significant amount of processing at each node and smaller transmission sizes than JPEG images, as shown for the data encryption application in Figure 9. The high energy consumption of nodes implemented with GPPs results in the network collapsing after only 200 requests. As expected, FGPA-based nodes perform slightly better, and the small-scale reconfigurable implementation introduced here maintains network function even longer. While ASIC-based nodes would consume even less energy, they would not have the necessary flexibility. In both network examples, small-scale reconfigurability provides the highest efficiency (and resulting fidelity) while still being capable of enabling the necessary parameter adaptations.

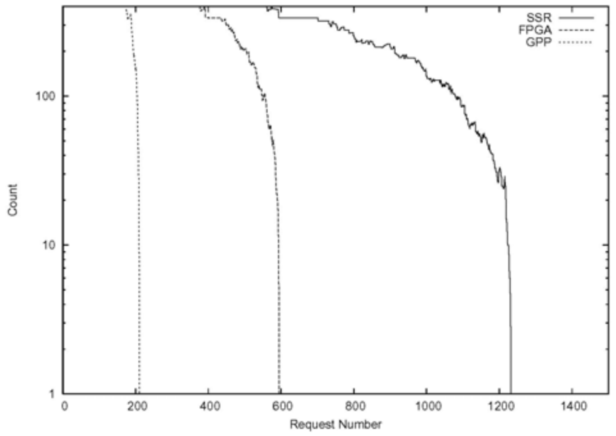


Figure 9: Impact of excessive flexibility on data encryption network lifetime with same energy consumptions assumptions as in Figure 8.

V. Conclusion

We have shown that for wireless sensor networks to maintain efficiency and utility, they must have dynamic adaptation abilities to deal with unpredictable and changing application requirements and to efficiently manage limited resources. In particular, the energy-limited nature of distributed nodes often requires the network to dynamically tradeoff fidelity, security or latency for network longevity. Changing algorithm parameters such as JPEG compression ratios or the number of encryption rounds, and using adaptive aggregation are necessary for the network's longevity while still providing its desired functionality. In addition, the information known about a network's application, scenario and deployment is limited at node design time, requiring a network to adapt to its configuration post-deployment. During operation, node failures and mobility often require dynamic topology adaptation to maintain network efficiency and utility.

The flexibility required for dynamic adaptation does not come without a cost. The software-based processing hardware commonly used for adaptation is significantly less energy efficient, is more expensive in large quantities, and has lower performance than inflexible ASICs. We described a heterogeneous small-scale reconfigurability design technique that provides flexibility with ASIC-like efficiency by finely integrating fixed and reconfigurable logic. Simulations revealed that nodes implemented using this technique achieve significant energy savings and a corresponding increase in network longevity.

Acknowledgments

This work is supported in part by the National Science Foundation under CCR-0105626, NSF CAREER (CCR-0092945) and NSF ITR (EIA-0205327) grants.

References

[1] T. He, J. Stankovic, C. Lu, T. Abdelzaher, "SPEED: A stateless protocol for real-time communication in sensor

networks," *International Conference on Distributed Computing Systems*, May 2003.

[2] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, J. Anderson, "Wireless sensor networks for habitat monitoring," *ACM International Workshop on Wireless Sensor Networks and Applications*, September 2002.

[3] X. Zeng, R. Bagrodia, M. Gerla, "GloMoSim: A library for parallel simulation of large-scale wireless networks," *Proceedings of the 12th Workshop on Parallel and Distributed Simulations*, May 1998.

[4] D. Panigrahi, C.N. Taylor, S. Dey, "A hardware/software reconfigurable architecture for adaptive wireless image communication," *International Conference on VLSI Design*, 2002.

[5] C. Intanagonwiwat, R. Govindan, D. Estrin, "Directed diffusion: A scalable and robust communication paradigm for sensor networks," *Mobile Computing and Networking*, August 2000.

[6] W.R. Heinzelman, A. Chandrakasan, H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," *Hawaii International Conference on System Sciences*, January 2000.

[7] C. Intanagonwiwat, D. Estrin, R. Govindan, J. Heidemann, "Impact of network density on data aggregation in wireless sensor networks," *International Conference on Distributed Computing Systems*, November 2001.

[8] Y. Yao, J. E. Gehrke, "The Cougar approach to in-network query processing in sensor networks," *Sigmod Record*, Vol. 31, No. 3, September 2002.

[9] S.R. Madden, M. Franklin, J. Hellerstein, W. Hong, "TAG: A Tiny AGgregation Service for Ad-Hoc Sensor Networks," *Annual Symposium on Operating Systems Design and Implementation*, December 2002.

[10] L. Hu, D. Evans, "Secure aggregation for wireless networks," *Workshop on Security and Assurance in Ad Hoc Networks*, January 2003.

[11] S. Coleri, M. Ergen, T.J. Koo, "Applications and OS: Lifetime analysis of a sensor network with hybrid automata modeling," *International Workshop on Wireless Sensor Networks and Applications*, 2002.