Tenet: An Architecture for Tiered Embedded Networks

Ramesh Govindan Eddie Kohler Deborah Estrin Fang Bian
Krishna Chintalapudi Om Gnawali Ramakrishna Gummadi Sumit Rangwala
Thanos Stathopoulos

Abstract

Future large-scale sensor network deployments will be *tiered*, with the motes providing dense sensing and a higher tier of 32-bit *master* nodes with more powerful radios providing increased overall network capacity. In this paper, we describe a functional architecture for wireless sensor networks that leverages this structure to simplify the overall system. Our *Tenet* architecture has the nice property that the mote-layer software is generic and reusable, and all application functionality resides in masters.

1 Introduction

Over the last five years, sensor network research has seen significant advances in the development of hardware devices and platforms, and in the design of services and infrastructural elements such as routing, localization, and time synchronization. Deployed systems, however, have lagged behind. Existing deployments are small- to medium-scale continuous data acquisition systems in which all sensor data is collected at a central location. This is far from the vision enunciated in early sensor networks work, in which sensor networks incorporate significant in-network processing for energy efficiency. Yet, attempts to move these deployments closer to the vision have foundered.

Our experience with sensor deployments has convinced us that the problem is with the vision. The constraints on programming low-power, mote-class systems—sensing uncertainty, wireless communication vagaries, and limited energy, processing, and memory—are difficult enough to handle on their own; yet the "Application Specific" principle [6] on which much sensor research is based suggests that each application must tackle these problems *combined with application-specific data fusion constraints*. This is leading us to develop systems that are exceedingly complex, unmanageable, and not re-usable. A new architecture is needed.

This paper discusses the architectural foundations of sensor networking. Our focus is on the *functional* architecture, the principles that, based on cost and complexity arguments, state where functionality should reside in a network. Our arguments are modeled after the end-to-end principle [15], which states how functionality should be placed in data communication networks. We call our principle

> **The Tenet.** Multi-node data fusion functionality and complex application logic should be implemented only in a tier of relatively high-powered Stargate-class nodes, which we call *masters*. The cost and complexity of implementing this functionality in motes outweighs the performance benefits of doing so.

The tiered embedded networks built on this principle, which we also call Tenets, contain both small-form-factor motes and Stargate-class masters. Tiered organizations have been discussed before [20]; our contribution is to simplify the architecture by explicitly limiting mote functionality. Motes contain sensing and actuation functionality and enable infrastructure-less instrumentation of physical spaces and artifacts, while masters are free of energy constraints and provide increased network and computational capacity, enabling large-scale deployments. All mote sensor data is routed to computational elements running on masters, or users and databases attached to masters. Motes are tasked by applications running on masters, and can implement simple logical elements such as thresholds and compression, but any further computation takes place only on masters.

Excluding multi-sensor fusion and complex application logic from motes will have two advantages: first, the application runs in a less resource-constrained environment, reducing development cycles and and improving overall system robustness; and second, the principle makes it possible to conceive of a *generic* mote layer that need be implemented once, and reused for a variety of applications. The disadvantage—a potential loss of efficiency—is a small price to pay for increased robustness and manageability.

We emphasize that the development of Tenet does *not* supplant research on mote-class devices. Motes are essential for low-cost dense sensing, and ongoing research on software architectures for the motes [1] and on various

mote subsystems, such as medium access, time synchronization, and localization, will remain highly relevant for Tenets.

Tenet opens up several novel research directions. One important area is the development of a generic master-to-mote interface that can be used by several applications. Another is the design of robust subsystems necessary for an operational Tenet: a robust routing system, reliable delivery of data between masters and motes, effective congestion control for high data-rate applications, low-overhead network monitoring, and automated networking management and tuning. The design of these subsystems can leverage masters, and the perspective they have of the mote network, for simplicity and efficiency.

The Tenet architecture can greatly accelerate the development of applications, and hence the adoption of this highly-promising technology. This paper discusses the Tenet architectural framework, and briefly discusses how it can simplify the design and development of sensor network applications.

2 The Tenet Architecture

In this section, we first briefly review the progress of sensor networks research so far, then describe and justify the architectural principle guiding the design of Tenets. We then describe the Tenet architecture in a little bit more detail, and discuss the relevant research challenges. We conclude this section by outlining the implementation of two qualitatively different applications on the Tenet architecture.

2.1 Motivation

Recent progress in sensor network research and development has been oddly nonuniform. Several groups have made major advances in hardware development, leading to two commonly-available classes of sensor platform: inexpensive *motes*, such as Crossbow's Mica series and similar devices from Telos and Dust Inc.; and so-called gateway nodes, which we call *masters*, such as Crossbow's Stargate. Masters have roughly an order of magnitude more computational power, memory, and wireless communication bandwidth than motes. Moreover, the community's research output has been impressive and wide-ranging, from lower-layer services such as MAC, routing, localization and time synchronization [5, 14] to higher-layer services and subsystems such as network programming [8].

Yet actual sensor network deployments [2, 16, 17, 19] have lagged behind this cutting edge. First-generation deployments are largely *continuous data acquisition systems*, where data from every sensor is collected at a central location. Such systems typically employ a clustered architecture, in which a master node is the head of a cluster of motes, and master nodes are connected to each other via a high-speed wireless backbone. Sensor data from each mote is transmitted multi-hop to the nearest master, and thence to a back-end database for storage.

Why this disconnect between research and deployment? An in-depth examination is beyond the scope of this paper, but we believe the answer is in the architecture. Most sensor network research has accepted, and worked within, an architectural principle the community articulated early on. In 1999, we expressed this principle as follows:

> **Application-Specific** Traditional networks are designed to accommodate a wide variety of applications. We believe it is reasonable to assume that sensor networks can be tailored to the sensing task at hand. In particular, this means that intermediate nodes can perform application-specific data aggregation and caching, or informed forwarding of requests for data. This is in contrast to routers that facilitate node-to-node packet switching in traditional networks. [6, Section 2]

Put simply, this principle hasn't stood the test of time. Our experiences with small-scale sensor network deployments over the past five years have convinced us that it needs wholesale revision. As with Active Networks [18], the application-specific mote vision provides significant opportunities for research, but the resulting systems are too complex to deploy and maintain. We now argue that *an architecture that pushes complex application-specific logic to the motes will lead to fragile and unmanageable systems*. While motes are essential to enabling low-cost dense sensing, programming on the motes is subject to constraints along many dimensions: limited memory, processing, and energy, together with communication vagaries and environmental uncertainty. Applications developed for the motes will have to respect these constraints, which means that application developers will need to be exposed to these constraints, and will need to manage resources in order to satisfy these constraints. This results in long lead times for application development, or fragile systems that need to be manually engineered in order to work in different environments. Systems designed this way will be error-prone and inflexible almost by necessity.

Of course, some of the challenges in these deployments have resulted from technological transients: platform immaturity and lack of any significant experience with systems embedded in harsh environments. Regardless, we think our argument will continue to hold. Mote constraints, such as environmental uncertainty and energy, are

either fundamental, or will require a technological revolution (*e.g.* miniaturized fuel cells) to overcome. Furthermore, our deployment experience has re-emphasized the fundamental importance of *robustness* and *manageability* for sensor networks. Sensor networks must survive arbitrary failures, and must give users and administrators insight into problems occuring within the network, since harsh deployment environments will necessarily induce new failure modes. In-network application processing only adds to the possible failure modes, further complicating an already difficult problem.

2.2 An Architectural Principle

Since systems based on the Application-Specific principle cannot achieve the robustness and manageability required for large deployments, we need a new architectural principle to guide our designs. The focus of this principle should be the reduction of complexity.

This paper discusses an architectural principle that can greatly simplify sensor network application development. Our principle applies to tiered embedded networks containing both motes and master-class systems. Again, we call this principle

> **The Tenet.** Multi-node data fusion functionality and complex application logic should be implemented only on masters. The cost and complexity of implementing this functionality in motes outweighs the performance benefits of doing so.

Tiered embedded networks built according to this principle we likewise call *Tenets*. To our knowledge, the Tenet principle has not been explored in the literature before. The rest of this section examines its assumptions and implications.

Tiered Networks Most existing and planned deployments, including the James Reserve habitat monitoring network, the Great Duck Island network, and the Extreme Scaling network, consist of two tiers of nodes, motes and masters. Masters have significantly higher communication capacity than motes (*e.g.*, an 802.11x radio), and can be engineered to have significant sources of energy (*e.g.*, a large solar panel and/or heavy-duty rechargeable batteries). This leads to two *fundamental* advantages for tiered networks over similarly-sized flat networks of motes. First, the more powerful radios of the masters can improve the overall capacity of the network. Second, a hierarchical network can be relatively easily engineered to constrain the network diameter; with wireless link loss rates in the 5–30% range (or higher) [22], a large diameter mote network will have a vanishingly small likelihood of packet delivery.

Thus, future large-scale deployments of wireless sensor networks can safely be assumed to contain both motes and masters (Figure 1); typical deployments will have 1–2 orders of magnitude fewer masters than motes. Tenets leverage masters to offload complex application-specific tasks from the motes. Since masters can be engineered to have energy, they will have one or more orders of magnitude higher processing power, memory, and communication capacity. This less constrained environment is far easier to program robustly. Furthermore, removing application-specific functionality from motes makes them, in turn, *generic*, facilitating mote reuse and economies of scale.

The need for hierarchy in sensor networks has been pointed out before [9, 21]. Prior research on heterogeneous sensor networks has described routing techniques that incorporate heterogeneity [7, 12, 13], and has examined how careful placement of a few highly-capable nodes can improve network lifetime [20]. However, Tenets are, as far as we know, the first attempt to leverage master functionality into a significant reduction of mote complexity.

Finally, the Tenet principle may simplify the development of sensor network systems that incorporate limited mobility, such as NIMS [3]. In particular, NIMS mobile components will employ Stargate-class devices, and can be treated as Tenet-style masters; limiting communication between NIMS mobile nodes and motes along Tenet-style lines should make NIMS more robust and deployable, just as in stationary networks.

Data Fusion Tenets also constrain multi-node data fusion functionality to be placed on masters. Such functionality can conceivably be generic, such as a generic tracking or beamforming service. Many existing multi-node fusion algorithms are clustered: the algorithm first needs to determine which motes have relevant data, then dynamically elect a fusion node (a cluster head), and finally route data to that node. The Tenet principle does not require us to rethink this algorithm structure. Rather, we suggest that masters form natural fusion points; since Tenets will usually be engineered for small network diameter, a master is likely to be found within a few hops of each mote. Happily, the resulting implementations will not require dynamic leader election or node discovery mechanisms at the mote layer.

The tradeoff is, of course, reduced efficiency, since sensor data needs to be routed from motes to the nearest master. There are several ways to mitigate this loss of efficiency. First, Tenet motes can implement *generic*, *local* sensor data processing, such as compression, transformation, or thresholding; these operations can significantly reduce transfer sizes. (As we discuss below, Tenet motes always forward data verbatim: no mote will alter data produced elsewhere.) Second, no mote will be far from

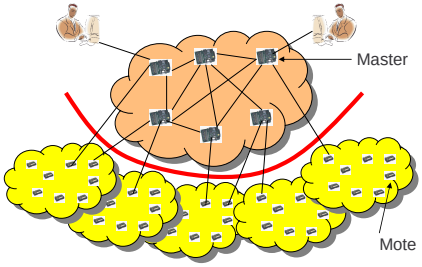

Figure 1: A tiered embedded network

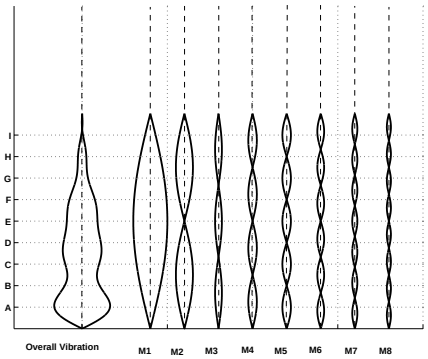

Figure 2: Vibration modes of a beam

a master, since many deployments will be engineered to have bounded network diameters for fairly fundamental reasons. Finally, the existence of masters can enable more aggressive management of mote energy resources: applications can adaptively adjust thresholds to limit the number of motes responding to an event; and, the system can adjust duty-cycling rates for motes not currently active.

Discussion Our arguments borrow heavily from the literature on Internet architecture, and the Tenet itself shares some similarities with the Internet's end-to-end principle [15]. Both principles discuss the placement of functionality within the network, and in both cases the rationale for the principle lies in the tradeoff between the performance advantages obtained by deeply embedding application-specific functionality and the cost and complexity of doing so. However, the end-to-end principle is slightly stronger than the Tenet, since it is based on a fundamental argument that is somewhat independent of technological trends, rather than a technological assumption that seems likely to hold true for the foreseeable future. But although a revolutionary advancement in battery technology or energy harvesting could call the Tenet into question, systems built today according to the predominant Application-Specific principle will certainly be less robust than Tenets.

2.3 Design Principles

The Tenet architectural principle constrains the design space for sensor network architectures, but is not itself an architecture. This paper presents not just the Tenet, but a coherent sensor network architecture built around it. This architecture is based on the Tenet itself and the four additional *design principles* described here.

The first principle pertains to the network-layer topology of a Tenet.

> **Addressability.** Any master in a Tenet can communicate with any mote or master in that Tenet, as long as there is (possibly multi-hop) physical-layer connectivity between them. Furthermore, any mote in a Tenet can communicate with at least one master in that Tenet, unless no master is reachable.

This principle helps enforce high network robustness and a relatively simple programming model. For example, imagine that a mote A is connected one-hop to a master M, but could be connected to a different master, M2, via three hops. The addressability principle requires that, if M dies, M2 will learn about and be able to address A, and vice versa. The requirement to support master-to-master communication allows, but does not require, the construction of distributed applications on the masters. Addressability requires much less of motes, however; a mote must be able to communicate with *at least one* master (assuming the network is not partitioned), not all masters, and mote-to-mote connectivity is not required. This is by design, and greatly simplifies mote implementations. We expect Tenet motes to communicate with masters, not with each other.

The form of this communication is determined by the second design principle.

> **Asymmetric Task Communication.** Any and all communication from a master to a mote takes the form of a task. Any and all communication from a mote is a response to a task; motes cannot initiate tasks themselves.

Here, a "task" is a request to perform some activity, perhaps based on local sensor values; tasks and responses are semantically disjoint. This principle restates the Tenet principle in concrete form, stating that Tenet motes communicate passively and masters actively. It disallows, for example, coordinated distributed triggering within the mote tier. Consider the case where a cloud of motes sends a message to a base station only when an aggregate sensor value passes some threshold. This requires examining more than local sensor values, and requires motes to task

each other. Essentially, motes may only be tasked by masters to collect data from specified sensors, or to actuate attached devices such as cameras or structural exciters.

Within these bounds, however, there is considerable flexibility, which we plan to explore as future work. For example, tasks might request immediate, periodic, or randomized data collection; results might be sent back uncompressed, sampled, aggregated (as an average, say), or transformed (using an FFT, say); and communication might be contingent on some sensor threshold. This general definition of a task will allow applications to program the network to be more energy-efficient. As the technology evolves, and improvements in battery capacity will permit more processing and memory resources to be added to motes, applications will be able to leverage these resources. This capability will allow increased network lifetimes or (equivalently) will allow applications to extract more "work" from the network for a given lifetime. The necessary requirement is that any multi-node or tasking functionality must be implemented on masters. Thus, the Tenet architecture does not contradict the need for in-network processing that prior sensor networks research has generally assumed. Rather, it precisely prescribes what kind of in-network processing can be performed where in a Tenet.

Practical Tenets will, of course, support management operations on the motes, such as determining the status of a task or of the mote itself, altering parameters of sensors and actuators, and reprogramming the motes themselves. The communication-is-tasking principle simply states that these operations must be modelled as tasks. Furthermore, in some cases, a sensor or exciter may be directly connected to a master; we model this as a master with a logically connected mote.

The third principle further defines what tasks may request of a mote.

> **Task Library.** Motes provide a limited library of generic functionality, such as timers, sensors, simple thresholds, data compression, and FFT transforms. Each task activates a simple subset of this functionality.

The task library is similar in intent to a processor or virtual machine [10, 11] instruction set, except that we expect the task library instruction set will not be Turing-complete: infinite loops, for example, will not be expressible. (This limitation simplifies the construction of robust mote software—for example, there's no need to worry about runaway scripts.) A task library that simultaneously simplifies mote programming, simplifies master and application programming, and provides maximum efficiency is a key piece of the Tenet architecture. Luckily, we expect to gain considerable leverage from concurrent work on the SNA architecture [1], which focuses on software interfaces and modularization for mote-class devices.

The fourth, and final, principle simply states the yardsticks by which we will measure success.

> **Robustness and Manageability.** Robustness and manageability are primary design goals.

Robust networking mechanisms, which permit application operation even in the face of extensive failures and unexpected failure modes, are particularly important for the challenging environments in which sensor networks will be deployed. This goal has long been recognized as fundamental for networked systems; the specialized communication pattern in Tenets impose a different set of robustness challenges than those faced by more traditional networks, but also provide a structure that may facilitate better solutions. Manageability implies, for example, that tools in the task library must provide useful insight into network problems—such as why a particular sensor or group of sensors is not responding, or why node energy resources have been depleted far faster than one would have expected—and allow automated response to such problems. Ensuring manageability is another classic problem in networking for which there exists a large literature and well-known issues in wired networks. By elevating both robustness and manageability into high-level design goals—above even performance—we hope to make significant progress on these vital issues, which are some of the major roadblocks to large-scale deployments today.

3 Implementing Applications

The Tenet architecture will enable us to implement different applications without changing the mote tier, with application-specific code and data processing residing on the masters. We illustrate this by sketching how two qualitatively different applications would be implemented in a Tenet.

3.1 Triggered Imaging for Habitat Monitoring

Consider an in-situ habitat monitoring sensor network equipped with various types of sensors. In particular, assume that some sensor nodes are equipped with cameras. Given the relatively high energy cost of imaging, it is desirable to *trigger* one or more cameras in response to information gleaned from other sensing modalities. When to trigger a camera, and which set of cameras to trigger, is an application-specific operation that might depend on data from a number of sensors (whose identities may not be known *a priori*), or may be a more complex function

that might depend on the history of previously sensed values.

An implementation of this application-specific functionality on the motes would require the following steps. First, when one of the motes locally detects the possible need to trigger a camera (*e.g.*, because a temperature reading has exceeded a threshold), it needs to discover other motes that have also detected these events. Then, it fuses these individual event detections in an application-specific manner to decide whether a camera needs to be triggered. Finally, it needs to dynamically discover the locations of cameras, and determine which ones to trigger.

These steps are difficult to implement on the motes. Implementing dynamic discovery and data fusion on energy- and computation-constrained devices can be a challenge. In particular, in this application, the collection of motes that is logically related to a camera (*i.e.*, is within the camera's field of view) can be different from the network's physical and topological relationships, and the relationship will change over time as obstructions move in and out of the space between cameras and the sensing motes, and as changes in connectivity result in network topology changes.

In a Tenet, this application would be implemented as follows. Each master runs application-specific code. Collectively, the masters task all (or a subset, as necessary) of the motes to report back when their temperature readings exceed a threshold. Each mote conveys this reading back to its nearest master; the masters (perhaps with some coordination) fuse this data, determine which camera(s) to trigger, and task those cameras.

This implementation is fairly flexible. Given the computational and memory resources of the masters, the fusion decision can be made more sophisticated (*e.g.,* by using historical information, or by incorporating information from other modalities that detect obstructions) by simply re-programming the application on the masters. A sophisticated triggering algorithm can avoid false positives. Furthermore, the application can dynamically tune the sensor thresholds (if necessary) by simply re-tasking the motes. This can be used to trade-off accuracy for overhead as necessary. Finally, the application can dynamically choose to add new sensing modalities by tasking more sensors. In this application, a different sensing modality might be used to detect obstructions to a camera view, for example.

The Tenet implementation might lose some energy efficiency, since masters must be informed of temperature readings as well as triggered images. However, this is by no means assured, since the inter-mote protocols that would be required to support camera location and so forth might be more expensive over time.

3.2 Structural Damage Detection and Localization

Structural health monitoring, or SHM, aims to develop technologies and techniques that automatically detect, localize, and classify damage in large structures (ships, bridges, aircraft and buildings) [4]. We envision an SHM system consisting of hundreds of sensors and several tens of exciters (actuators). This system would periodically perform a set of tests on the structure to determine its structural integrity, and locate possible damage inside the structure. A typical set of tests would involve inducing forced vibrations at different points and analyzing the structure's distributed responses.

The structural response of a structure is often represented as the composition of several modes. A mode is a standing wave pattern (Figure 2) induced on a structure and is characterized by a modal frequency (one of the resonant frequencies of the structure), a mode shape (the spatial amplitude distribution of the structural vibration at that resonant frequency), and an attenuation (how fast the energy of the standing wave decays). Damage in a structure typically alters one or more of the modes of a structure. Mode detectability depends on sensing and actuation locations within the structure. For example, in Figure 2, a sensor placed at location E will never detect modes M_2, M_4, and so on, while E is a good place to detect mode M_1. Actuation at E will generate a dominant mode M_1 while it will not generate mode M_2. A *simultaneous actuation* in opposite directions at locations I and G will generate mode M_5. A real structure typically has several different modes of very complex 3-dimensional shapes.

Existing SHM techniques rely on collecting all the structural response data resulting from a set of actuations and performing a modal analysis on them. Modal analysis applies signal processing operations such as Fourier transforms (FFT), singular value decomposition (SVD), computation of ARMA parameters, computing auto-correlations, and so forth. Such modal analysis can detect the *existence* of structural damage by, for example, measuring changes in the frequency of one or more modes. It can also detect the *location* of the damage by measuring changes in mode shape. Sophisticated in-network modal analysis is simply out of reach of today's mote technology. As motes evolve, they may become more capable of performing complex computations, but the difficulties in successfully distributing modal analysis will remain.

A Tenet is a more natural architecture for this class of high-data rate applications. Application specific code on the masters would implement the structural tests by tasking sensors and exciters. In a typical test, one or more masters would coordinate and task motes as follows: command one or more exciters to actuate the structure at

one or more locations at a certain time; collect the structural response at a specified frequency (say 100 Hz) for a specified duration (say 1–2 seconds); then transmit the data back to a master. The masters would then to check whether the structure is damaged, possibly via collaboration. In the absence of damage, they would command the motes to sleep until the next scheduled test.

With improvements in mote technology, some of the modal analysis, specifically the generic signal processing steps, can be implemented at the motes. For example, a master can task a mote to compute the ARMA coefficients of the sensed data, instead of sending back time series. This will involve a relatively simple change to application code running on the master tier, while still avoiding involving the motes in complex coordination.

4 Conclusion

The two applications discussed above are qualitatively different in that they have different computational requirements, different sensing modalities and sensor data rates, and different time-scales of operation. Yet, we argue that *both* can be implemented by using *identical* software for the mote tier, and largely common infrastructure for the master tier. The goal of developing the Tenet architecture is an ambitious one. We envision sensor networks that are truly easy to design, program, deploy, and manage. Tenet mote-class systems will implement a fixed library of functionality, enabling mostly-hardware implementation and further reducing cost and size, and allowing true application-level sharing of mote infrastructures. The software tools we develop will make it easy to implement interesting multi-master applications. Tenet represents a large step towards our community's shared end goal—ubiquitously deployable sensor networking.

References

[1] A Network Architecture for Wireless Sensor Networks. `http://today.cs.berkeley.edu/SNA/`.

[2] The Extensible Sensing System. `http://www.cens.ucla.edu/~eoster/ess/`.

[3] Maxim A. Batalin, Mohammad Rahimi, Yan Yu, Duo Liu, Aman Kansal, Gaurav S. Sukhatme, William J. Kaiser, Mark Hansen, Gregory J. Pottie, Mani Srivastava, and Deborah Estrin. Call and response: experiments in sampling the environment. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 25–38. ACM Press, 2004. ISBN 1-58113-879-2.

[4] J. Caffrey, R. Govindan, E. Johnson, B. Krishnamachari, S. Masri, G. Sukhatme, K. Chintalapudi, K. Dantu, S. Rangwala, A. Sridharan, N. Xu, and M. Zuniga. Networked Sensing for Structural Health Monitoring. In *Proceedings of the 4th International Workshop on Structural Control*, Columbia University, NY, June 2004.

[5] Jeremy Elson, Lewis Girod, and Deborah Estrin. Fine-grained network time synchronization using reference br oadcasts. In *Proceedings of the Fifth Symposium on Operating Systems Desi gn and Implementation (OSDI 2002)*, Boston, MA, December 2002.

[6] Deborah Estrin, Ramesh Govindan, John Heidemann, and Satish Kumar. Next Century Challenges: Scalable Coordination in Sensor Networks. In *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking*, Seattle, WA, USA, August 1999. ACM.

[7] Wendi Rabiner Heinzelman, Anantha Chandrakasan, and Hari Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *HICSS '00: Proceedings of the 33rd Hawaii International Conference on System Sciences-Volume 8*, page 8020. IEEE Computer Society, 2000. ISBN 0-7695-0493-0.

[8] J. Hui and D. Culler. The Dynamic Behavior of a Data Dissemination Algorithm at Scale. In *Proceedings of the ACM Conference on Embedded Networked Sensor Systems*, Baltimore, MD, November 2004.

[9] V. Kottapalli, A. Kiremidjian, J. P. Lynch, E. Carryer, T. Kenny, K. Law, and Y. Lei. A Two-Tier Wireless Sensor Network Architecture for Structural Health Monitoring. In *Proc. of SPIE's 10th Annual Symposium on Smart Structures and Materials*, San Diego, CA, March 2003.

[10] Philip Levis and David Culler. Mate : a tiny virtual machine for sensor networks. In *ASPLOS-X: Proceedings of the 10th international conference on Architectural support for programming languages and operating systems*, pages 85–95. ACM Press, 2002. ISBN 1-58113-574-2.

[11] Philip Levis, David Gay, and David Culler. Active sensor networks. In *Proc. 2nd Symposium on Networked Systems Design and Implementation (NSDI '05)*, Boston, Massachusetts, May 2005. To appear.

[12] Stephanie Lindsey and Cauligi S. Raghavendra. PEGASIS: Power-Efficient GAthering in Sensor Infor-

mation Systems. In *IEEE Aerospace Conference, 2002*.

[13] Arati Manjeshwar and Dharma P. Agrawal. TEEN: A Routing Protocol for Enhanced Efficiency in Wireless Sensor Networks. In *15th International Parallel and Distributed Processing Symposium (IPDPS'01) Workshops*, 2001.

[14] Joseph Polastre, Jason Hill, and David Culler. Versatile Low Power Media Access for Wireless Sensor Networks. In *Proceedings of the ACM Conference on Embedded Networked Sensor Systems*, Baltimore, MD, November 2004.

[15] J. H. Saltzer, D. P. Reed, and D. D. Clark. End-to-end arguments in system design. *ACM Trans. of Computer Sys.*, 2(4):277, November 1984.

[16] Robert Szewczyk, Alan Mainwaring, Joseph Polastre, John Anderson, and David Culler. An analysis of a large scale habitat monitoring application. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 214–226. ACM Press, 2004. ISBN 1-58113-879-2.

[17] Robert Szewczyk, Eric Osterweil, Joseph Polastre, Michael Hamilton, Alan Mainwaring, and Deborah Estrin. Habitat monitoring with sensor networks. *Commun. ACM*, 47(6):34–40, 2004. ISSN 0001-0782.

[18] David L. Tennenhouse and David J. Wetherall. Towards an active network architecture. *Computer Communication Review*, 26(2), 1996.

[19] N. Xu, S. Rangwala, K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan, and D. Estrin. A Wireless Sensor Network for Structural Monitoring. In *Proceedings of the ACM Conference on Embedded Networked Sensor Systems*, Baltimore, MD, November 2004.

[20] M. Yarvis, N. Kushalnagar, Harkirat Singh, A. Rangarajan, Y. Liu, and Suresh Singh. Exploiting Heterogeneity in Sensor Networks. In *Proceedings of the IEEE Infocom*, 2005.

[21] M. Yarvis and W. Ye. Tiered Architectures in Sensor Networks. In Mohammed Ilyas (ed.), editor, *Handbook of Sensor Networks: Compact Wireless and Wired Sensing Systems*. CRC Press, July 2004.

[22] J. Zhao and R. Govindan. Understanding Packet Delivery Performance In Dense Wireless Sensor Networks. In *Proceedings of the ACM Conference on Embedded Networked Sensor Systems*, Los Angeles, CA, November 2003.