Management of networked sensor systems

ALEXANDROS KOLIOUSIS and JOSEPH SVENTEK
University of Glasgow

The ready availability of integrated circuits for sensing (MEMS), processing and wireless communication has resulted in burgeoning interest in the design, implementation, deployment, and operation of environmental sensor networks. The maintenance and control of such systems is essential to ensure efficient use of resources for appropriate information gathering and processing; since most of these networks must (of necessity) operate in an unsupervised manner, these systems must also recover from partial failure or changes in the sensed environment. Currently, management functionality is inextricably intertwined with application, operating system, and/or networking components. Additionally, despite the existence of management frameworks for well-resourced, wireline networks, there is no reason to believe a priori that such frameworks will apply to the resource-constrained environments of typical environmental sensor networks. We survey network management functionality in existing sensor networks (focussed on battery-powered, wireless sensors that are static or move infrequently). This survey yields a set of dimensions for the different categories of management functionality implemented in the surveyed systems: deployment, diagnostics, application software, bandwidth, energy, and security management. We then discuss how use of such a classification scheme can aid the structured development of the management aspects of future sensor network systems.

1. INTRODUCTION

Sensor networks exemplify a natural evolution of distributed systems. Miniature hardware devices with complex functionality have inspired a diverse set of monitoring applications; environmental and habitat data acquisition is an exemplar application domain of sensor monitoring systems. Deployed in remote, usually inaccessible areas, sensing applications can generate information at a constant rate (time-driven), or in reaction to environmental and network stimuli (event-driven) or user-initiated queries (query-driven).

A sensor system is a large-scale distributed system. A set of N sensor nodes operate collaboratively towards a common goal in a resource and time constrained environment. Given an initial energy budget, memory budget, CPU budget, and network bandwidth budget, the system must ensure an acceptable level of fidelity during its lifetime. Network and system management is a distributed activity involving the invocation of a set of management functions destined to guarantee this fidelity, as dictated by the functional and non-functional requirements of the system.

Sensor systems must be adaptive to changing conditions in the sensed area of in-

The authors wish to acknowledge the support of the Engineering and Physical Sciences Research Council (EPSRC) under grand EP/C014774/1.
Authors' address: Alexandros Koliousis, Joseph Sventek. Department of Computing Science, University of Glasgow, 17 Lilybank Gardens, Glasgow, G12 8RZ; e-mail: {koliousa,joe}@dcs.gla.ac.uk.

terest, both at an application and system level. Sensor systems consist of adaptable and re-configurable components. Management components monitor the behavior of the sensor system and, if necessary, adapt the system. The management system orchestrates these mechanisms via policy. Non-functional requirements are expressed either as resource and real-time constraints, or service objectives. The management system also provides an interface for the re-configuration or update of application components.

A separation between policy and mechanism is necessary to guarantee the flexibility of a sensor system. The diversity of constraints and objectives for a given application render them impossible to capture under a single policy framework. Currently, management functionality is scattered and tangled, usually duplicated, across the network protocol stack (e.g. energy management services), making policy modification a daunting task. On the contrary, management components should be independent of layer implementation; they can be accessed and invoked via interfaces, available to arbitrary components of the system. A set of management mechanisms with distinct functionality is sufficient to translate, monitor, and enforce the system policy.

Management mechanisms typically coexist with application software components, sharing the same system resources. Management systems introduce an additional overhead to the sensing application, in terms of energy, communication, computation, and storage resource usage. A user expects, in return, significant gains in terms of resource savings and application operability. As an example of such controversy, Maté [Levis and Culler 2002] trades off the additional CPU overhead to satisfy the requirement for application adaptability and updates. However, if updates are infrequent, better performance is expected by a monolithic binary image.

A management system must be application-cooperative, that is it must be optimized to support the unique requirements of each application. The requirement for application cooperativeness, however, does not coincide with the desire for application-independent management systems. First, sensing applications must be reactive to changes in the environment they operate; the management system should provides the means to adapt the application protocols when required. Second, management functionality must be available before the initial deployment and after catastrophic application failures [Tolle and Culler 2005].

Current projects (e.g. [Martinez et al. 2004]) require connectivity with external wireless or wireline networks for the display and visualization of application and management information. Existing Internet technologies are sufficient to elaborate on such integration. However, the *intelligence* of the system, fragments of code representing the application and management logic, must still reside within the sensor network. The processed, and possibly aggregated, information can be either propagated directly to the end-user, via one or more sink nodes, or stored in the network for subsequent retrieval.

It must be possible for the management system to transfer responsibility to an appropriately authorized system, if such control is asserted in the proximity of the network. We define access points in a network as nodes through which an external entity can access management information, and act upon the sensor network. Network management via Web services [Ramanathan et al. 2005; Song and Kim 2005]

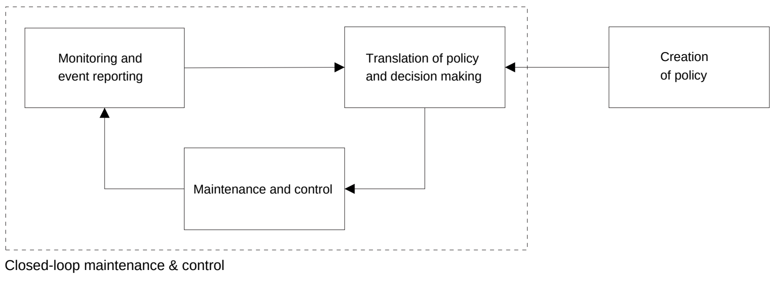
Monitoring and event reporting
Translation of policy and decision making
Creation of policy
Maintenance and control
Closed-loop maintenance & control

Fig. 1. Maintenance and control loop of a sensor system.

and SNMP variants [Ruiz et al. 2003] has been proposed in the literature for the remote management of sensor networks[1].

Internet technologies may be sufficient for macromanaging the sensor network (e.g. specify a high-level policy); however, micromanagement must be performed *in situ*. The OSI management framework, the most common manifestation of which is the Simple Network Management Protocol (SNMP), does not completely capture their unique management requirements. Sensor systems usually operate unattended and thus, by necessity, are autonomic. The degree of autonomy is determined by the number of orchestrating components in the sensor system. A (subset of) sensor node(s) is enriched with management components to drive closed-loop control of the sensor system (autonomic management), relaxing the need for network operators (Figure 1).

This article is a survey of the management mechanisms proposed thus far for sensor networks. We revisit the proposed management systems for sensor networks, extract their functional components, and classify them in a attempt to identify *what* are the management requirements in a sensor system and *how* they are addressed. Six functional areas enable the reader to navigate through the sensor network and system management space. Deployment, diagnostics, application software, bandwidth, energy, and security management typify the sensor network management areas of concern and form the basis for the realization of a sensor network and system management framework.

2. DISTRIBUTED MANAGEMENT PARADIGMS

Sensor nodes are expected to operate unattended during their deployed lifetime. This requirement for autonomy, in conjunction with the large number of sensor nodes, mandates the transition from centralized to distributed management architectures. There are three management paradigms for sensor systems based on the level of delegation of management responsibility: flat, hierarchical, and federated

[1]For example, BOSS (Bridge Of the SensorS) nodes [Song and Kim 2005] implement the Universal Plug and Play (UPnP) protocol to enable the remote management of sensor networks; the Sympathy debugger [Ramanathan et al. 2005] provides a Web interface at the sink for configuring system-wide parameters or querying the status of individual nodes; in the MANNA management system [Ruiz et al. 2003], access is provided via agents, processes that reside on sensor nodes and present information about the status of (possibly, a subset of) nodes.

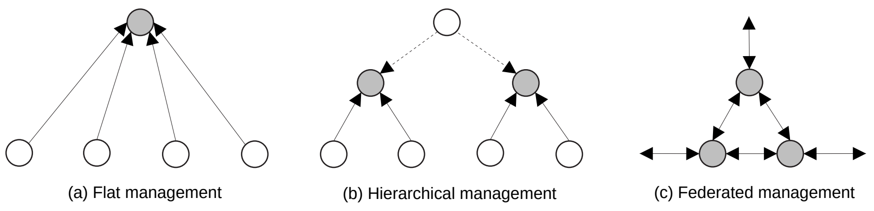

Fig. 2. Flat, hierarchical, and federated management architectures for sensor networks. Grey nodes indicate management-aware sensors. In hierarchical architectures, management responsibility is delegated to a small subset of nodes. Sensor nodes in federated paradigms collaborate to complete a management task.

management paradigms (Figure 2).

Flat management paradigms are characterized by sensor nodes being unaware of the management tasks. They simply provide an interface for a manager entity to access, or modify, the status of a node. Processing and analysis of the collected management information is assigned to a centralized manager node, responsible for the orchestration of the system. The polling phase can be either scheduled, or initiated explicitly, by injecting a query to the network. Exemplar flat management architectures include the Nucleus [Tolle and Culler 2005], Sympathy [Ramanathan et al. 2005], and MANNA [Ruiz et al. 2003] management systems.

In hierarchical management paradigms, the processing of management tasks is confined to a small number of sensor nodes, termed the sub-managers of the system. Hierarchies are defined in terms of power (delegation by capability) or network topology (delegation by domain). Delegation by capability assumes a heterogeneous network, where certain nodes are more powerful in terms of communication, energy, computation, or memory resources [Younis et al. 2003]. Such nodes are usually carefully placed in the network to contribute to the performance of the sensing application.

In delegation by domain, sensor nodes are grouped into clusters based on their location in the network. Among the nodes in each cluster, one is selected to serve as the cluster-head. The role of cluster-head may be rotated over time among different nodes to balance the resource consumption within a cluster [Savarese et al. 2001; Cerpa and Estrin 2004]. Usually, it is assumed that cluster-heads are within the communication range of the manager. Because all communications within a cluster are routed through its cluster-head, they latter has the necessary knowledge to perform certain management tasks.

Transmission of 1 bit of data consumes the same amount of energy as the execution of $\sim$1000 instructions [Levis and Culler 2002]. As the number of sensor nodes within a cluster grows, we need to push management responsibility towards the end nodes, rather than polling them. A federated management paradigm is characterized by the management awareness of the end nodes. Sensors make management decisions locally, based on their own information, or they collaborate with their neighbors to reach a common consensus about the status of their domain. Management tasks can be delegated to the end nodes as computational processes, management goals, or policies.

Mobile code for sensor networks has been suggested to address issues of application adaptability and updates [Marsh et al. 2004; Liu and Martonosi 2003; Levis and Culler 2002]. Such technologies for sensor networks enable the development of mobile-agent-based approaches to sensor network management. Dissemination of code has been also proposed [Jaikaeo et al. 2001], where aggregation logic is propagated to nodes via scripts. Management tasks can be also described as goals, or policies, using a high-level specification language [Frank and Römer 2005; Hull et al. 2003]. Such rules can be pre-installed, or dynamically disseminated to sensor nodes at runtime. Depending upon the management architecture selected, policies can be disseminated via flat, hierarchical, or federated logic. Finally, application of economic theories to management, e.g. market-based programming [Mainland et al. 2005], has also been found to lead to federated paradigms.

3. AREAS OF CONCERN

Management responsibility can be divided into conceptual areas based on the purpose and scope of management. Within an area, responsibility manifests in the management environment by a subset of the total system components and their interactions. Our survey of sensor management systems establishes a categorization of management components according to their functional behavior in six areas of concern:-

a. Deployment management;
b. Sensor network diagnostics;
c. Application software management;
d. Bandwidth management;
e. Power management; and
f. Security management.

In a similar manner, the Open System Interconnection (OSI) management framework [ISO 1989] divides management and its standards into five functional areas: fault, configuration, accounting, performance, and security management. The OSI functional areas aimed to assist the standardization process of the OSI management model for large distributed systems. At the heart of the OSI environment is configuration management – it exercises control over the network elements based on monitoring information (e.g. performance, diagnostics, or accounting information).

The OSI management model consists of an information model based on an object modelling paradigm, a communication model based on agent and manager roles, and a set of specific function standards for the monitoring and control of network elements. Information flow is defined in terms of operations and notifications based on the pull model[2]. The OSI management environment was initially manifested in flat management paradigms. Since then, the management community has focused on hierarchical management architectures[Goldszmidt and Yemini 1995].

A sensor system is an example of a large distributed system. Given an initial configuration, a network of constrained processors cooperate to accomplish a desired

[2]Push mode notifications are supported via SNMP traps, but the unreliable nature of the notification delivery mechanisms prevents most common uses of these notifications.

task. By design, sensors do not exhibit byzantine behavior, and thus the system does not require accounting for resource usage. The management environment must allocate, negotiate, and if necessary redistribute, the global resources of the system (i.e. energy and bandwidth) amongst sensors to ensure availability.

Sensors must operate in many different types of environments in the absence of human operators. Control functionality (i.e. configuration management) indicates the action, not the requirement, to assert autonomy in a system after the initial deployment and while awaiting maintenance. As such, a sensor can configure its state given partial or complete knowledge of the network.

The close affinity of sensor requests and responses forces management in the aggregate. Collection and dissemination protocols optimize resource consuption by incorporating arbitrary in-network aggregation logic to sensors according to the present application model (e.g. for merging or piggybacking management data onto environmental measurements). Consequently, it is difficult to impose a communication model amongst management processes.

The needs of sensor management systems are best met by the push communication model. Given the regularity and redundancy of management requests, sensors retain memory via schedulers or triggers and broadcast data or events towards orchestrating components, at regular time intervals or asynchronously. On the contrary, data polling requires an explicit request to initiate a message exchange. The push paradigm favors the autonomous nature of sensors, while it conserves energy and bandwidth resources. The pull model can complement a sensor management protocol when control is asserted by a human operator.

The N-layer management of the OSI framework enables the analysis of the management functionality at the n^{th} layer in isolation. However, optimality in a sensor system is achieved by flattening the communication stack through co-design, yielding cross-layer optimization [Su and Lim 2006]. The deployed system must manage the n^{th} layer via policy with layer-independent management functionality.

The MANNA management model [Ruiz et al. 2003] attempts to apply the OSI management model to sensor systems by incorporating sensor network functionality as a new dimension to the OSI functional areas. It is difficult to reason about management requirements in such a complex, monolithic abstract structure. Furthermore, the MANNA architecture follows the OSI management model, thus inheriting its strong dependencies upon information and communication models that we have seen to be incommensurate with sensor network requirements.

We employ a bottom-up methodology to identify the management requirements in a sensor system. Our approach is more pragmatically focused than the OSI management model. We survey existing management solutions and identify six areas of concern for sensor management: deployment, diagnostics, application software, bandwidth, power, and security management. Note that the proposed categorization can be projected to the OSI functional areas, if required.

Overlapping management functionality (*how* the management requirements are addressed, in contrast to *what* are they) is expected among the different dimensions. The existing management components are discussed along three different axes: their support for application models (i.e. time-driven, event-driven, or query-driven), their selected management paradigm, and their resource requirements.

3.1 Deployment management

Sensor nodes are densely deployed throughout an area of interest. The deployment of the sensors can be either random or planned. The sensor network is formed *in situ.* Sensors probe the environment for ambient phenomena and transmit their measurements, via multi-hopping wireless communication channels, to an appropriate authority, usually a sink node; they can also process or aggregate data streams to postpone resource depletion. These are the *sensing*, *routing*, and *processing* operational modes of a sensor node.

Networked sensors usually operate and manage in the aggregate [Tolle and Culler 2005]. Queries, or commands, are usually disseminated to every node of the network. Otherwise, a subset of nodes is identified based on their geographical position or their data measurements. For high density networks, the system can operate partially while providing sufficient connectivity for the reliable collection and dissemination of information. The modes of these operational sensors can then be configured accordingly to ensure adequate coverage of the deployed area, while removing any unnecessary redundancy. On the other hand, low density networks require (weak) node mobility to ensure coverage and connectivity within an area of interest. For example, the Virtual Force Algorithm (VFA) [Zou and Chakrabarty 2004] performs a one-time movement to evenly distribute sensor nodes in an area after an initial random deployment.

Deployment management functions are executed during the pre-deployment, deployment, or post-deployment phases of the sensor system. Pre-deployment configuration refers to information introduced to sensor nodes prior to deployment. The deployment, or initialization, phase is dominated by the presence of management traffic. Post-deployment configuration is required due to changes in the network. For example, nodes or communication links may fail, nodes may change their position, or new nodes may be introduced to the system.

3.1.1 *Location discovery.* The *in situ* deployment of wireless sensor systems establishes a relationship between sensor nodes and the physical world. The user of a sensing application must be able to relate certain events or measurements to spatial positions. Localization systems are designed to identify the spatial relationships between sensor measurements and physical location. We focus on localization systems for randomly deployed sensors. For planned systems, the position of the nodes can be asserted *a priori.* Furthermore, we assume that not all (and possibly none of the) sensor nodes are equipped with the ability to accurately determine spatial position, such as a Global Positioning System (GPS) receiver.

The coordinate system of a sensor network can be relative with regard to the deployed area or a particular node[3]. Relative systems can be translated to absolute coordinate systems given the known position of a subset of nodes, the *anchor nodes.* The precision of the localization system is proportional to the number of anchor nodes. Location discovery systems impose an additional financial, energy, communication, and computation cost to the sensing application.

The basic requirement for any location discovery system is a measurement of the distance (or range) between any two nodes of the sensor system. The local coordi-

[3]A sensor node, e.g. a cluster-head, may assume the coordinates $(0, 0, 0)$.

nates are derived by solving a 3-D triangulation problem. A localization system can be either range-based or range-free [He et al. 2005]. Range-based systems require accurate distance measurements, based on the received strength, angle of arrival, or time of arrival of a signal [Savarese et al. 2001]. On the contrary, range-free systems [He et al. 2005] have no requirement for absolute distance measurements; they rather utilize the known position of anchor nodes and estimates (via hop counts) of the node distance from them.

Systems implementing a flat management paradigm assume a global, centralized computational engine. Range measurements are propagated to a central node, where the localization problem is solved. Afterwards, the position information is propagated to every node of the network. Such approaches consume a considerable amount of the system resources. Cooperative ranging approaches [Savarese et al. 2001], based on the federated paradigm, have emerged to overcome the limitations of centralized systems. The localization problem is solved locally, at the node level, based on information from neighboring nodes. The resulting coordinates are then propagated to the network. If present, the procedure is initiated from anchor nodes. Ultimately, the network converges to a global coordinate space. Further iterations of the distributed algorithm can improve the accuracy of positions. However, the refinement phase adds to the overall cost of location discovery. As an example, the TERRAIN [Savarese et al. 2001] algorithm requires at least four anchor nodes to converge and twenty-five iterations to reduce position error to 5%.

The APIT localization algorithm [He et al. 2005] is an exemplar range-free counterpart of cooperative ranging approaches; computation is fully distributed and is performed at each node, based on message exchanges with neighbor nodes. The APIT algorithm utilizes signal strengths to estimate its position within a triangle of anchor nodes. A node can be inside one or more triangles (thus, it confers with neighbor nodes to identify them); the maximum overlapping area forms a triangle in which the sensor node assumes to be positioned in the center of gravity.

It must be possible to establish a coordinate system in the complete absence of anchor nodes. A beaconless, distributed positioning algorithm has been proposed [Iyengar and Sikdar 2003]. A master node collects the distance estimates from its neighboring nodes and establishes a local coordinate system within a cluster. A node claims to be master if no other candidates exist within a given time interval. Master nodes sharing two or more nodes at the border of their clusters collaborate until they converge to a global coordinate system. The convergence time of the algorithm is inversely proportional to the density of the sensor network. A similar, federated, GPS-free algorithm has also been proposed [Capkun et al. 2001]; rather than electing a master node, each sensor assumes it is the origin of the coordinate system and solves the triangulation problem locally; the relative coordinates are again propagated to the entire network until the algorithm converges.

Figure 3 illustrates a classification of location discovery systems. A centralized, global solution to the localization problem is an expensive operation. If the positioning task is delegated towards the sensor nodes the cost is distributed among them; the communication overhead and convergence time become now proportional to the density of the network. Hierarchical structures have been found to reduce the overall cost significantly, in contrast to purely distributed approaches. The esti-

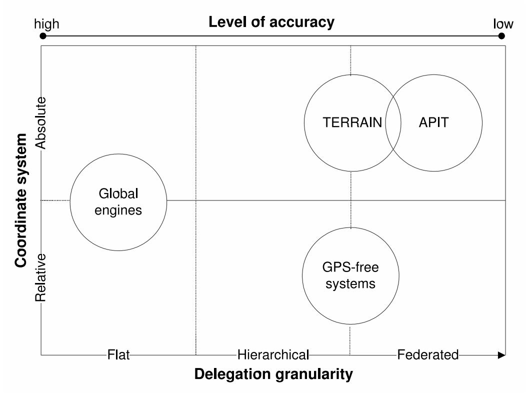

Fig. 3. A classification of location discovery systems.

mation error (or accuracy) of federated paradigms is subject to further iterations of the localization algorithm. The error margin in federated range-free systems (e.g. APIT) depends on the range estimate formula.

3.1.2 *Topology maintenance and control.* Topology management functions coordinate the sleep transitions of all nodes to conserve resources, while ensuring a connected topology, with sufficient capacity for the reliable propagation of data to the sink nodes [Schurgers et al. 2002]. Topology maintenance and control requires the collaboration of neighboring nodes to ensure connectivity within a small region of the network. It is strongly intertwined with routing protocols for sensor networks. Topology control protocols may utilize location information (geography-informed), routing information (topology-based), or other management and application logic.

In the GAF protocol [Xu et al. 2001], nodes collaborate to form a logical grid based on their physical location. A grid cell contains one or more sensor nodes. Nodes within a grid cell can communicate directly with any node in a neighboring cell. Nodes within a cell collaborate to elect a single node to be active. The discovery phase is repeated periodically to balance the load among all nodes within a grid cell. The "one active node per cell" rule implies a linear increase of the network's lifetime as the node density within a grid cell increases. For cells with a single node, it is assumed that mobile sensors join the cell to actively participate in routing.

The SPAN protocol [Chen et al. 2001] does not require any physical location information; it rather utilizes routing information to ensure a connected, multihop topology. SPAN opts to preserve the original capacity of the network by designating certain nodes as coordinators. A sensor node decides, based on information collected from its neighboring nodes, to participate in the network as a coordinator if the minimum distance between any two of its neighbors exceeds two hops. Thus, the algorithm imposes an upper-bound limit to the number of transmissions within the network.

STEM [Schurgers et al. 2002] is a topology control protocol for event-based networks. It assumes that no communication is required unless an event is generated.

STEM prolongs the sleep period of the radio while idle. Upon detection of an event, a node sends a wakeup signal to the next neighboring node to route through the event. The process is propagated until the data message arrives at the sink. STEM assumes the existence of a dual frequency radio to eliminate the interference of wakeup and data messages. Finally, it assumes that no periodic management traffic exists in the network.

The ASCENT protocol [Cerpa and Estrin 2004] is independent of location and routing information. ASCENT reacts to packet loss; if the number of packets lost exceed a predefined upper threshold, new nodes join the network to ensure connectivity. During the initial phase of the algorithm, nodes wait for neighbor announcement messages. If the density of their neighborhood exceeds a threshold, nodes enter the passive state; otherwise, they become active. Passive nodes eavesdrop on the communication medium and become active if they detect high packet loss or receive a help message from an active sensor.

3.1.3 *Role assignment in wireless sensor networks.* Active sensors cycle among the three operational roles, i.e. sensing, processing, and routing, to evenly distribute the workload among them according to their capabilities, thus extending the network lifetime, while providing sufficient coverage of the sensed area. Surplus nodes will result in redundant information; this may improve the accuracy of measurements, but it degrades the network performance by increasing the traffic load. It can result in high packet loss due to collisions, and consequently, energy wastage. Data redundancy can be reduced either by in-network processing, or by turning off unnecessary nodes. Network configuration aims to assign specific roles to sensors to meet the application requirements while preserving the resources of the system.

Role assignment is supplementary to topology control management, and vice versa. While topology control protocols can ensure the connectivity of the network, additional active sensors may be required to provide complete coverage of the area. Furthermore, given a localized phenomenon, a partially connected network may be sufficient for the routing of measurements to the sink nodes. On the other hand, role assignment algorithms may utilize topology information, e.g. by invoking a neighbor discovery process, to efficiently configure the network.

The multihop sensor network management problem [Perillo and Heinzelman 2003] is a joint optimization effort of role assignment and route selection. Sensor nodes are logically divided into feasible sets, combinations of nodes capable of meeting the application's requirements. The objective of the problem is to schedule the different feasible sets to maximize the network lifetime. For each feasible set, the algorithm must determine its operation time. Given the multi-hop nature of sensor networks, routing policies also affect the network lifetime, and thus they are introduced to the problem as constraints. Currently, the optimization problem is solved by a centralized computational engine.

Similarly, the Sensor Network Life Problem [Berman et al. 2004] also aims to maximize the network lifetime by defining sensor cover sets, each of them providing sufficient coverage, either partial or complete, of the network. However, the problem has been addressed both for centralized, and federated architectures. In the latter case, the system assumes three states for a sensor node, active, idle, or vulnerable.

A vulnerable sensor may become active, if insufficient coverage is provided by the network, or idle, if sufficient coverage is provided by nodes with higher energy budget. Essentially, a vulnerable state is an undecidable state. Once a node enters a vulnerable state, its state is propagated to the entire network and roles, active or idle, are reassigned.

The generic role assignment [Frank and Römer 2005] framework is a programming abstraction for the pre- and post-deployment configuration of the network. A generic role specification, described as rules for accepting a specific role, is generated and propagated by the manager of the system to every node. The rules are evaluated at each node, based on local and neighboring node information, and one or more roles are allocated optimally among them. The proposed system addresses problems of coverage, clustering, and in-network aggregation by assuming a set of roles for each problem instance.

Once the specification is disseminated by the manager, the role assignment algorithm evaluates the rules based on property tables maintained at each node. Properties are propagated proactively after a local, randomized timer expires to avoid traffic bursts. Random delays are also introduced during rule evaluation to avoid simultaneous role assignments. The procedure is repeated during the network's lifetime in response to property changes (e.g. in the battery level), node additions, or device failures until a fixed-point configuration is found, where each node assumes a stable role for a period of time.

In the Self-Organizing Resource Allocation (SORA) approach [Mainland et al. 2005], a sensor network is modelled as a virtual market where nodes are self-interested agents that generate goods by taking appropriate actions, i.e. sensing, routing, or aggregating. Prices for each action are distributed by a central manager node. Each node selects an operational mode that maximizes its utility function. They adjust their behavior based on payment feedback they receive from the sinks. Even if sensor nodes are self-interested, an equilibrium arises that balances the actions among nodes.

Moving towards distributed architectures, configuration management can be expressed using high-level languages (e.g. [Frank and Römer 2005; Mainland et al. 2005]). Figure 4 illustrates a categorization of role assignment algorithms based on the semantic richness of management logic. A managed object represents an exported attribute of the sensor node, e.g. residual energy level or number of neighbors. The polling of these objects can be distributed to individual sensor nodes, enabling the manager to specify tasks using an abstract computational logic. By specifying the semantics of a management task using high-level goals (or policies) the degree of autonomy and adaptability of the management system increases significantly.

3.2 Sensor network diagnostics

A management system must be able to determine the correctness of the sensing application during its deployed lifetime. It requires the propagation of network diagnostic information to a designated node for processing and analysis. Failures in sensor networks are routine rather that sparse events. Diagnostic functions must indicate system and device failures or resource depletion. Furthermore, the system must be able to log certain events for real-time or post-mortem analysis. Fault

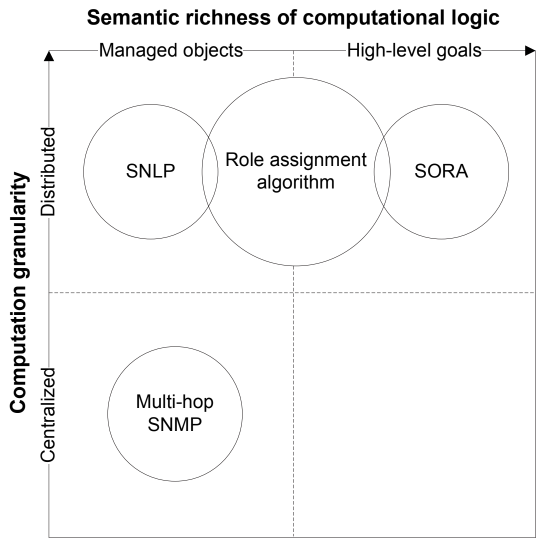

Fig. 4. A classification of role assignment systems in wireless sensor networks.

management is essential in every sensing system.

Information gathering, or monitoring, in sensor networks is a strongly centralized procedure. The manager node –usually a sink node– becomes a single point of traffic concentration. Essentially, diagnostics and application monitoring mechanisms anticipate the same network behavior. Thus, it is important to reduce the communication overhead imposed by the diagnostics management mechanisms over the sink nodes in favour of the sensing application.

The response implosion problem in centralized management architectures has been addressed [Jaikaeo et al. 2001]. Three operations have been proposed, based on probabilistic, time-based, and in-network aggregation approaches. The Adaptive Probabilistic Response (APR) operation assumes a hierarchical sensor network. It assigns a response probability on every node in the network. Given a non-uniform distribution of nodes in each cluster, the probability is the number of responses required from a cluster, divided by the number of nodes in the cluster.

Time-driven approaches use heuristics to determine the delay period before a response is injected in the network. Given the assumption that packets from nodes with the same distance from the manager will eventually collide, the system introduces an extra delay; i.e. the system defines a virtual queue where packets are propagated sequentially. Furthermore, as packets from distant nodes are less frequent, the manager can increase the sending rate for nodes towards the edge of the network.

In-network aggregation reduces the amount of management traffic around the manager station. The diffused computation operation [Jaikaeo et al. 2001] proposes an agent-based approach to network aggregation; the manager of the network propagates a script containing the aggregation rules. However, although management functionality is pushed to the network, nodes become resource-aware, not management-aware. Diagnosis still needs to be performed at the manager level, based on the collected aggregated information. Furthermore, not all management functions can act in the aggregate. Finally, localization of a failure may be difficult, or impossible. For example, the monitoring system in [Zhao et al. 2003] initiates scans to a specific area to locate a failure.

Computing aggregates for monitoring the health of the sensor system has also been proposed in [Zhao et al. 2003]. In this approach, aggregated data, or digests, are continuously propagated to the manager node. Digests are not initiated by the manager. They are rather pushed from the edge sensor nodes upstream. Piggybacking the monitoring traffic onto periodic data-link or application layer traffic results in a minimization of the management overhead.

The STREAM algorithm [Deb et al. 2003] is a distributed algorithm for sensor topology retrieval at multiple resolutions. In contrast to probabilistic or in-network aggregation models, the STREAM algorithm finds a minimum set of nodes to reply to a query, i.e. a topology discovery request, given a prerequisite resolution of the network. The motivation behind STREAM is that not all nodes are required to respond to a query in a given situation; some data may be redundant. STREAM finds an approximate solution for an NP-complete coloring problem. It uses four colors, white, black, red, and blue. White nodes represent inaccessible nodes. Black nodes respond in the aggregate, on behalf of red nodes. A blue node waits for a neighboring node to be colored black; otherwise, it becomes black itself.

Monitoring of sensor networks can be active or passive. Active monitoring assumes the existence of periodic traffic (e.g. periodic beacons or keep-alive messages). Active monitoring results in an implicit detection of a failure; absence of traffic indicates an error in the network (e.g. [Ruiz et al. 2004]). On the contrary, passive monitoring assumes no management traffic unless an error occurs. The manager then receives an alarm explicitly indicating the failure.

The Sympathy [Ramanathan et al. 2005] management system exploits both active and passive (eavesdropping) measurements to detect failures in a sensor network. Rather than considering data quality, Sympathy is based on data quantity. The Sympathy system assumes that periodic communication will occur even in event-based networks. In particular, the sink node expects a specific amount of traffic from the network. An alarm is triggered if less traffic is generated. Detection and localization of a failure is based on connectivity (e.g. routing tables), flow (e.g. packets transmitted), and node (e.g. uptime) metrics. It identifies three possible causes of failure: self, path, and sink.

Passive measurements are always preferable because they introduce no additional bandwidth overhead. However, since active measurements are essentially required to determine the root cause of a failure, passive measurements may be redundant. Sympathy utilizes the merits of passive monitoring. However, implementing a centralized paradigm, active monitoring is also required to receive metrics from distant nodes. Thus, passive monitoring requires the delegation of diagnostic logic towards the end nodes.

We discuss the requirement for distributed, federated management paradigms for system diagnostics using a counterexample. Fault management for event-based sensor networks [Ruiz et al. 2004] using the MANNA architecture assumes a hi-

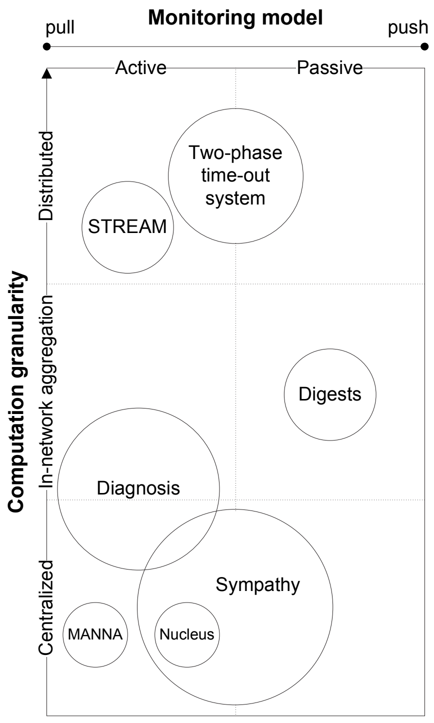

Fig. 5. Sensor network diagnostics systems. Diagnostic logic can be processed either centrally, or while in transit (in-network aggregation), or locally, at the node level.

erarchical architecture of the network. Cluster-heads act as SNMP-agents; the Management Information Base (MIB) of the cluster-head contains also information about its subordinate nodes. A manager issues SNMP-like `get` commands to query the status of the network nodes. It is assumed that lack of communication indicates an error in the network. However, such assumption can be false since environmental or network inference (e.g. a physical obstacle or a packet collision, respectively) may disrupt normal communication. We wish to push detection mechanisms closer to the actual occurrence of the failure. If a detection is verified *in situ*, then the manager can be notified.

A two-phase timeout system [Hsin and Liu 2002], that utilizes local node collaboration to reduce false alarms, is an example of a federated management architecture. The proposed system utilizes active neighbor monitoring. Each node is enriched with a decision making process, to increase its confidence regarding the occurrence of an alarm (and thus decrease the probability of a false alarm). The idea of a two-phase timeout control is that once the an event is detected (absence of traffic from a specific node during the first timeout period), the reporting node first consults its neighbors (second timeout period) about the accuracy of the event.

Finally, logging mechanisms are required for the storage of diagnostic information locally, in the memory of a sensor node. Events and alarms can be stored for future or post-mortem analysis. The Nucleus management system [Tolle and Culler 2005] provides a logging mechanism for debugging information during the execution of the sensor application. Logs, i.e. `fprintf` commands, are encoded to reduce the memory requirements of the mechanism. Logs can be queried by the manager in real-time for system diagnosis. A logging mechanism, however, is not a mere storage procedure. Non-critical alerts or events can be stored, and possibly aggregated, to avoid unnecessary transmissions. For example, messages can be stored until they can be piggybacked to application data to reduce the communication overhead.

Diagnostic management systems are summarized in Figure 5. In the pull monitoring model, collection of diagnostic information is explicitly (query-driven) or implicitly (time-driven) initiated by the manager of the system. Passive systems push management data towards the manager station in the occurrence of an event or an alarm.

3.3 Application software management

Sensor systems operate in a versatile, evolving universe driven by changes in the sensed environment, application requirements, and network status; the system must be reactive both to external and internal stimuli. Sensor nodes must accept re-configuration commands, software updates, or complete binary images to address the needs of such a demanding environment.

Software adaptability and updates for monolithic systems are impractical for two reasons. Intuitively, it is infeasible to capture the runtime application requirements *a priori*; even so, it is a memory demanding process. Secondly, the update process requires the propagation of a new, compiled binary image of the system to each node. The GRATISplus [Kogekar et al. 2004] modeling environment has been suggested to overcome these inherent difficulties by enabling the re-configuration of monolithic sensor systems.

A wide spectrum of applications can be derived by a common set of components [Levis and Culler 2002]. The design space of such applications is captured by formally modeling the components, their interfaces, and their interactions using the GRATISplus modeling environment. Application requirements are modeled as constraints to the system. A state transition in the operation space designates a reconfiguration command.

Sensor parameters (e.g. remaining energy level) are monitored locally and propagated to a centralized station at regular time intervals. If a violation is detected, then the system updates the application constraints and a new, valid system design is generated. The new and previous systems are compared to generate a set of commands for the transition to the new configuration. Commands are propagated to each node. At the node level, the application is stopped, rewired, and restarted. Reconfiguration is based on switching components embedded inside the system to rewire application components.

The proposed solution has certain limitations. First, new components cannot be introduced into the system; such a system update would require the propagation of the complete binary image. Furthermore, the system provides limited adaptability; re-configuration is restricted within the limited set of pre-installed components. Second, evaluation and verification of the system is resource expensive. The proposed management system is weakly centralized, based on a hierarchical architecture.

We focus on distributed software management architectures, implemented on

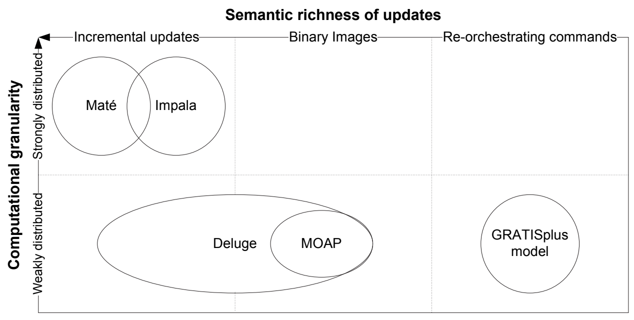

Fig. 6. A classification of application software management systems.

agent-based execution environments. Moving towards a federated management paradigm, the communication overhead of software management is reduced significantly. In addition, agent-based operating systems have inherent support for application adaptability and updates. Figure 6 shows a classification of application software management systems based on weakly, and strongly distributed paradigms.

Maté [Levis and Culler 2002] is a VM-based programming environment for sensing applications, based on TinyOS. Programs in Maté are broken into one or more capsules. Each capsule can fit in a single TinyOS packet. Maté uses an event-based programming model. Execution begins in response to an event. Each event is associated with one of the four types of code capsules: message send (for routing a packet), message receive, timer, and subroutine. The latter is used to express more complex, user-defined programs. During their dissemination, each capsule contains a version number; the most recent version is installed and executed.

Impala, a non VM-based, modular architecture for software management has been proposed [Liu and Martonosi 2003]. This programming model is also event-based. However, events are propagated to the adapter or the updater component of the architecture. Thus, in contrast to Maté, the application can continue running even if the update process is in progress; no switching command has been issued yet. Impala assumes the presence of multiple application protocols on board. It introduces an autonomic system for application adaptability. A management process (the adapter) monitors periodically the state of the sensor and, if a switching rule is satisfied, it activates the new application protocol. By incorporating the adaptability functionality at the sensor level, nodes become more robust to device failures.

Code distribution requires a reliable dissemination protocol that guarantees message delivery. Maté has been implemented over Trickle [Levis et al. 2003], a code propagation and maintenance protocol. Trickle uses periodic broadcast messages within a time interval to inform other nodes about its current software version. If similar transmissions by neighboring nodes exceed a predefined threshold, the node remains quiet. For networks with no time synchronization, the Trickle protocol suffers from increased maintenance traffic. To address this problem, the protocol defines a listen-only period before it starts to gossip, in order to suppress any unnecessary metadata transmissions. If a node has an older version of the software or, respectively, a node has a new software version, the code is broadcast. The infection rate of the Trickle algorithm is determined by the time interval during which metadata are exchanged. A short time interval means rapid propagation but also high maintenance traffic.

Impala implements a three-phase handshake protocol for the dissemination of module updates. Similar to Trickle, in Impala nodes first advertise their latest software version. However, if a software transmission is required, Impala will initiate explicitly a session with a currently updated node by sending a request, rather than listening for a broadcast message from the latter.

Deluge [Hui and Culler 2004] and MOAP [Stathopoulos et al. 2003] represent a family of epidemic protocols for the quick, reliable dissemination of large binary images over a high density network. The Deluge protocol is also built on top of Trickle. If a transmission is required, Deluge ensure that a bi-directional channel exists before transferring the data. The data object is divided into fixed-size pages to enable efficient incremental updates, in contrast to MOAP which requires a complete image transmission. Only nodes that have a complete version of the last page may advertise and propagate data to neighboring nodes. If a node requests an update, it uses the latest advertisement to select the sender. Both Deluge and MOAP implement a NACK-based approach to detect packet loss.

3.4 Bandwidth management

Control over the bandwidth distribution among different traffic flows is essential to guarantee an acceptable level of network Quality of Service (QoS). Bandwidth management in wired networks is implemented both at the end nodes (end-to-end congestion control) and the intermediate routers (e.g. fair queuing algorithms). Intuitively, this implies the delegation of bandwidth management responsibility to each sensor node; besides generating traffic locally, they act as relays to route through traffic. We investigate bandwidth management mechanisms that address issues of congestion control, bandwidth allocation, and traffic classification.

Congestion occurs when the offered load, the amount of traffic generated by the individual sensors, exceeds the available network capacity. It results in increased packet loss, and thus energy wastage, degrading the overall performance of the application. As the size of the network increases, congestion collapses may become more frequent. While congestion can be detected *a posteriori* by collecting diagnostic information, congestion control is required to preserve the real-time network performance.

It is difficult to define an upper bound for the required capacity of a sensing application in advance. There are two reasons for this difficulty. First, route through traffic can be aggregated by nodes when appropriate. Second, occasional bursts of events increase the transmission rate of sensor nodes. Thus, over-provisioning of network resources is an inadequate solution [Ee 2005]. Furthermore, end-to-end congestion control mechanisms, such as TCP, are not applicable to sensor networks since the bursty nature of sensor network traffic results in artificial window sizes, while introducing an additional communication overhead due to the large number of end-to-end acknowledgments [Hull et al. 2003].

Hop-by-hop flow control has been suggested in [Hull et al. 2003] as an alternative to end-to-end congestion control. It exploits small, link-layer packets to send synchronous NACKs (negative acknowledgments) to the sender if the queue size at the receiving node exceeds a predefined threshold. The sending node, upon receipt of a NACK, stops transmitting packets and eavesdrops the receiver's transmissions. If two packets are transmitted by the receiver, communication is resumed. Similarly, the Argus [Ee 2005] management mechanism performs congestion control via admission control; the receiver node defines the rate at which packets are injected into the network. The sender eavesdrop on the communication channel for piggybacked information into data packets. It reduces the injection rate when the queue size of the receiver is found to exceed a specified threshold.

It must be possible for the sensing application to allocate bandwidth to different data flows. Argus implements a variation of Weighted Fair Queuing, the Extended Epoch-based Proportional Selection (EEPS) algorithm, to effectively distribute bandwidth to downstream nodes. In particular, a sensor node, the parent node, receives a bandwidth request, in units of flow, from each of its downstream nodes, the child nodes. By dividing its local transmission rate by the total units of flow requested, the node obtains its per-unit-flow generation rate. The latter is compared with the node's parent flow rate, and the flow rate used when the queue size exceeds a threshold; the smallest value is propagated to the child nodes. The inverse of the flow rate defines the epoch length. During this interval, the number of packets must not exceed the available capacity. Argus maintain a FIFO queue for each sender-destination pair, to ensure that bandwidth is allocated to all flows.

Energy-aware QoS routing [Akkaya and Younis 2003] is another approach to bandwidth allocation. Two types of traffic are defined, real-time and non real-time traffic. Each node maintains a separate queue for each packet type. The gateway calculates centrally the optimum bandwidth allocation rate for any link in the system. The value is broadcast to all sensor nodes. The amount of bandwidth dedicated to every link is optimized to provide best effort QoS to non real-time traffic, while meeting the requirements of all real-time traffic flows in a congested network.

Finally, the management system must provide mechanisms to prioritize critical data flows. In contrast to bandwidth sharing approaches, such as Argus, the manager of the system must be able to favor certain flows of traffic. A classification mechanism based on delay priority semantics has proposed [Hull et al. 2003]. Packets are forwarded based on their priorities. In case a packet must be dropped, it would be a low-priority packet. Packet priorities are defined based on a rule system. Rules are enforced by the manager of the system and propagated to each node. Given the type of packets, their value, and a set of node attributes (e.g. location), a rule is evaluated to yield the desired reception rate and the importance of the data stream.

Currently, bandwidth management functionality is implemented at the network and transport layer. This provides an acceptable communication abstraction level for network management mechanisms. For example, the Argus implementation is independent of the underlying MAC protocols. However, in order to support flow priorities, slight modifications are required to link-layer protocols. In particular, the system must ensure that high priority flows win any contention competition for the wireless medium.

	Congestion control	*Bandwidth allocation*	*No. of classes*
[Hull et al. 2003]	Hop-by-hop flow control	Rule system	n
Argus	Admission control	EEPS	2
Energy-aware QoS routing	–	Class queuing model	2

Table I. A summary of bandwidth management systems.

Table I summarizes the systems discussed in this section with regard to how they address the bandwidth management challenges –i.e. congestion control, bandwidth allocation, and traffic classification– in a wireless sensor network.

3.5 Power management

Sensor nodes are usually battery operated and thus energy constrained. They must enter a sleep mode when possible to preserve energy and, consequently, extend the network lifetime. The hardware components of a sensor node (i.e. the processor, memory, analog-to-digital converters, and radio) have different power modes. If k_i are the possible sleep states of a component i, then the possible sleep states of a sensor node are $\prod k_i$ [Sinha and Chandrakasan 2001]. Power management functions aim to schedule the transitions among these power states to conserve energy.

A shift between two states introduces an energy, memory, and latency cost. For example, turning off the processor requires the storage of its state into memory. Switching the radio on consumes a certain amount of energy and time until the radio is active; while waiting for the radio to be ready, one or more packets can be dropped. In many cases, greedy ON-OFF schedules may be inappropriate for certain applications. For time-driven applications, where application and management traffic are generated periodically, the ON-OFF schedule is trivial. However, in the occurrence of non-periodic events the schedule must be defined appropriately.

Power management functionality spans the different layers of the protocol stack. In particular, it has been incorporated in data-link, network, or application layer protocols. For example, the sensor-MAC (S-MAC) [Ye et al. 2002] and Timeout-MAC (T-MAC) [van Dam and Langendoen 2003] protocols provides power management functionality at the data link layer. They define a schedule for the radio to enter a sleep mode periodically to conserve energy. When an internal timer expires, the radio awakes and listens for transmission requests. Schedules are broadcast by a node, the synchronizer, to immediate neighbors, the followers. If a node does not receive a schedule for a specific period of time, it assumes the role of the synchronizer.

Topology control protocols (Section 3.1.2) assume that a node may remain in a sleep state for longer periods of time. The power management functionality of such protocols is complementary to the one provided at the link layer. For example, they may utilize location or topology information to completely turn off a node if it is considered unnecessary for the sensing or routing task.

Moving up the protocol stack, lower level functionality may become redundant. Furthermore, intelligent management decisions can be made, and thus power is controlled more efficiently. For example, the STEM protocol [Schurgers et al. 2002] turns off the radio completely until an event is detected. After processing, the node returns to sleep mode. However, transition to minimum power state may be

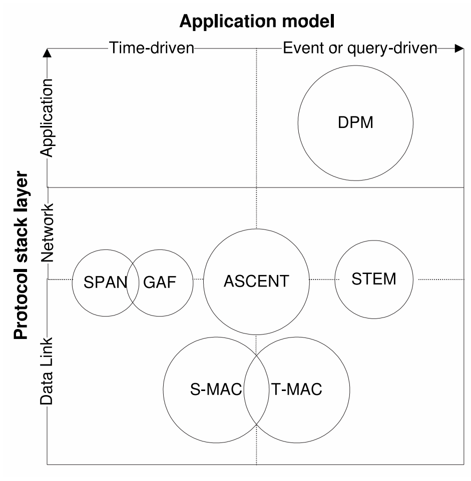

Fig. 7. A classification of power management systems.

inappropriate if an event if likely to occur within the time interval to enter the sleep state, and become active again. Dynamic Power Management (DPM) [Sinha and Chandrakasan 2001] uses a probabilistic model to predict the occurrence of such events and adjust the sleep schedule accordingly.

Energy conservation has been the focus of various optimization problems for sensor networks [Mainland et al. 2005; Perillo and Heinzelman 2003; Younis et al. 2003]. In the problem formulation, the residual energy of the system serves as a contraint which is traded off against Quality of Service (QoS). However, power management functions are destined to preserve the remaining energy of the system, rather than react to energy loss.

Power management functionality must be made available, via service interfaces, to the different layers of the protocol stack and used appropriately, according to the requirements of the application. By liberating it from specific implementations, a power management function may utilize information available at different layers, e.g. network topology or sampling rate, to find an optimal state transition schedule.

Figure 7 illustrates a classification of power management systems. Topology control protocols are also discussed herewith, because their derived operational schedules aim to conserve the residual energy of the network. Ascending the protocol stack, the number of possible components, and their sleep states ($\prod k_i$), that the power management system can manage increases.

3.6 Security management

A sensor system, deployed in a hostile environment, must ensure a secure communication channel between any two communicating nodes to protect them against passive or active attacks. The multicast nature of radio communications underpins passive attacks, i.e. eavesdropping on the communication channel; an adversary can easily intercept and alter messages once he/she/it has access to the deployed area. Potential physical access to the deployed area also entails that the system must be resilient to node capture attacks, either random or selective, or node fabrication attacks. Finally, the system must make provisions against Denial of Service (DoS) attacks to sensor nodes.

Cryptographic keys play a central role in sensor network security. Resource limitations in sensor networks favour the use of symmetric key cryptography, although asymmetric, public key cryptography is also feasible [Watro et al. 2004]. Sharing a single key among all nodes is a poor security strategy; once a node is compromised, the entire network is exposed to an attacker. Currently, security management schemes randomly distribute a set of cryptographic keys among sensor nodes to prevent such exposure. Key management involves the invocation of functions for the generation, distribution, storage, revocation, and update of cryptographic keys in a sensor system. It is related with the pre-deployment, deployment, and post-deployment configuration of sensor networks.

A key or polynomial pool is generated offline by a centralized station. In the basic probabilistic scheme [Eschenauer and Gligor 2002], keys are randomly drawn without replacement from the pool and loaded into the memory of each sensor. The number of sampled keys is proportional to the probability that any two nodes share at least one cryptographic key. In polynomial pool-based schemes [Huang et al. 2004; Liu and Ning 2003; Moharrum and Eltoweissy 2005], the system generates a pool of bivariate t-degree polynomials. Polynomial shares are distributed among the sensor nodes; a secure communication link is established if two nodes have shares on the same bivariate polynomial.

After deployment, a sensor node seeks those neighbors with whom it shares at least one key. During the discovery phase each node broadcasts an encrypted list of key or polynomial identifiers to its immediate neighbors. Neighboring nodes establish the shared key, if one exists, with the broadcasting node; essentially, the key graph connectivity describes the topology of a secure cryptonet. A link exists if, and only if, two sensor nodes share a cryptographic key. To establish a secure link between two nodes within communication range but without a shared key, the system initiates the path key establishment phase. In this phase, one or more intermediate nodes, which share different keys, or polynomials, with each of the two nodes, are responsible for establishing a common key among them.

Grid-based key distribution schemes [Huang et al. 2004; Liu and Ning 2003] assume a structured, $m \times n$ grid topology of the network. They utilize deployment information in an attempt to increase the performance of the system. In terms of resilience to attacks, grid-based schemes guarantee that a pairwise key can be established between any pair of nodes if the number of compromised nodes does not exceed a threshold. By restricting the number of nodes within a grid cell, a system is perfectly secure against random or selective capture attacks [Huang et al. 2004]. The key discovery phase is repeated to establish connectivity among different cells. In terms of performance, by imposing a structure on the key distribution a node can determine if a pairwise key or polynomial is shared with another node directly and thus no additional communication overhead is charged to the system.

The key management schemes discussed so far are static; the key distribution process is executed once, during the initial deployment of the network. If a node is compromised, its shared keys with other sensor nodes must be revoked; secure

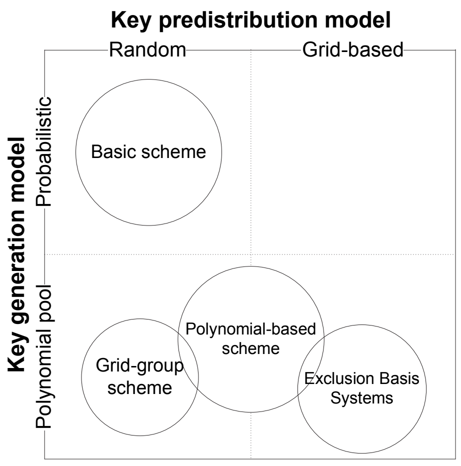

Fig. 8. A classification of key management systems based on their key generation and pre-distribution model. A key management system can be either static or dynamic, based on its key re-distribution model. Dynamic systems, e.g. Exclusion Basis Systems, can use either traditional or polynomial keys; their distribution can be either random or grid-based.

communication links are re-established based on the remaining pre-distributed keys. If the number of compromised nodes exceeds an upper threshold, the entire network is compromised. Dynamic key management schemes repeat the distribution process periodically if the lifetime of a key set expires or in response to a node capture attack [Moharrum and Eltoweissy 2005].

Key management is a core security management service for sensor network and is summarized in Figure 8. It serves as the basis for the authentication mechanisms in sensor networks. Given the aggregate nature of sensor networks, security mechanisms focus on message, rather than entity authentication. Encryption is an inadequate mechanism to ensure message integrity over an insecure communication channel. TinySec [Karlof et al. 2004] is a link layer security architecture that employs symmetric key cryptography for the authentication of sensor messages. It assumes a key management system for the reliable distribution of cryptographic keys. A packet has an encrypted checksum attached; the authenticity of a message is verified at each hop.

Employing a Public Key Infrastructure (PKI) for hop-by-hop authentication in a sensor network is impractical due to the resource limitations of sensor nodes. They may be unable to store and process key certificates or communicate directly with the Certification Authority (CA). However, hop-by-hop authentication is a security primitive upon which advanced security mechanisms can be built. The TinyPK architecture [Watro et al. 2004] introduces public key cryptography for the authentication of an external entity to the network via a centralized CA. Sensor nodes are unable to handle certificates and thus the CA's public key is pre-installed on every node. An external entity is authenticated to the sensor network via a PKI handshake, and vice versa. After successful authentication, a session key is generated to secure the message exchange.

Finally, the management system must provide an infrastructure for the logging, and reporting of security alarms to a designated entity of the system. Logging mechanisms can be used for auditing (e.g. a record of access attempts by an external entity) or post-mortem analysis. Security alarm reporting is essential to maintain the integrity of the system. For example, a node may report a security alert when the rate of authentication failures exceed a predefined threshold [Karlof et al. 2004]. The manager can locate the cause of the failure and take appropriate actions. We argue that the same mechanisms presented in Section 3.2 for sensor network diagnostics are adequate to support these security management requirements.

3.7 Summary

Components within a dimension address different management requirements and make different trade-offs against the application and network requirements. Currently, each dimension has been addressed orthogonally. A naive attempt to simply implement management functionality separately may result in colliding strategies. For example, a state transition schedule for power management [Sinha and Chandrakasan 2001] may collide with the ON-OFF schedule derived from the role assignment algorithm [Frank and Römer 2005]. Management mechanisms must be jointly considered to avoid such collisions in the configuration of sensing systems.

Deployment management functions exemplify the requirement for co-design in sensor management systems. The three services (localization, connectivity, and coverage) interact to assemble a network infrastructure for information gathering. As one ascends the protocol stack, more knowledge of the network is required to manage the higher levels, as thus more reliable solutions can be derived. For example, an optimal schedule to address both coverage and connectivity requirements of an application can be derived by utilizing information about the remaining energy or the topology of the network.

Diagnostic management is characterized by periodic transmissions (e.g. health beacons), with occasional bursts of traffic. Diagnostics are required to ensure the resilience of the sensing system when it enters a vulnerable state. Failure alarms or events must be accurate with minimum response delays. Energy and bandwidth management requirements are usually introduced to the system simply as budget constraints [Akkaya and Younis 2003; Perillo and Heinzelman 2003; Younis et al. 2003]. However, an efficient power management strategy can significantly increase the lifetime of the network; furthermore, the system may prioritize flows (e.g. alerts or software updates) to manage the timeliness or criticality of the network traffic.

Software management is dependent upon the programming and execution model of the sensing system. For example, code dissemination in Maté [Levis and Culler 2002] is based on code capsules, the program representation in this environment. Software management is dominated by metadata packets, exchanged between nodes until they reach a desired state. The convergence time, the time required until the network becomes useful again, is proportional to the number of nodes. A code dissemination protocol must ensure reliable code delivery, without saturating the network.

Finally, security management functionality is related to efficient key distribution and establishment among sensor nodes. Besides adding an additional cost to the application, in terms of resource utilization (e.g. memory or CPU usage), security strongly affects the pre-deployment, deployment, and post-deployment phases of a sensor system. As an example, the key distribution and establishment process can rearrange the connectivity graph of the network.

4. COLLECTION AND DISSEMINATION PROTOCOLS

An integral part of a sensor network management system is a *communication model* for the reliable collection and dissemination of management information and commands. Management data must be uniformly represented across the network, based on a *management information model.* A sensor node must provide an interface to the system for querying or modifying these data. A sensor management system must be able to manage in the aggregate; a point-to-point polling of sensor nodes, similar to the Simple Network Management Protocol (SNMP), could introduce an unacceptable overhead to the network.

The MANNA Network Management Protocol (MNMP) [Ruiz et al. 2003] for sensor networks is a variant of the SNMP protocol. MNMP conforms with the OSI management framework for distributed systems. The MANNA information model consists of a set of managed objects that represent the underlying network resources, their attributes, and interfaces. These objects form the Management Information Base (MIB) of the network. The manager communicates with an agent, a process that serves as an interface for reading or modifying the MIB data, via the `get` and `set` commands, respectively. To overcome the limitations of explicit polling, the MANNA architecture may place the agent process at the cluster-head level; end nodes push management data upstream to create a MIB abstraction for a group of nodes [Ruiz et al. 2004].

Collection and dissemination of management data in the Nucleus management system [Tolle and Culler 2005] is achieved via a collection tree construction protocol and the Drip protocol, respectively. Both protocols focus on reliable message delivery, rather than energy conservation. The tree construction process is initiated explicitly by one or more access points, or sinks, and is refined while the network is actively managed. An upper bound on the number of tree construction messages is set to minimize the communication overhead. Each node maintains a single best parent based on the link quality. The link quality is estimated based on the received signal strength.

The Drip protocol is built on top of the Trickle dissemination algorithm [Levis et al. 2003]. It uses periodic advertisements to ensure reliable message delivery to every node of the system. To address only a subset of nodes, the Drip protocol implements a naming component. Each message has three additional header fields, a destination address, a destination group, and a Time To Live (TTL) variable. Upon receipt of a message, the naming component is called to determine if the forwarded message is destined for the node. The message can be forwarded or dropped, if the TTL has expired.

Nucleus defines a query processing system to query or modify management data at the sensor node level. A sensor component exports a set of attributes, each associated with a canonical name. The attributes are organized by the developer of the application, rather than defining a static information model. Each attribute corresponds to a key in the data schema to reduce the size of queries and responses.

A management query retrieves a set of attributes which are compacted into a single message. Continuous measurements require the re-injection of the query to the system. The results are interpreted centrally, by a manager entity.

The SINA architecture [Shen et al. 2001] defines SQTL, a procedural scripting language used for querying, tasking, or reprogramming sensor nodes. Each node has a build-in interpreter for SQTL scripts. An SQTL message is delivered to every node of the network. To address a specific subset of nodes the message is wrapped in a header that indicates the receivers of the message. SINA introduces the notion of an associative spreadsheet as an information model for the system. Node attributes are stored in this spreadsheet. Rather than querying attributes based on logical (x, y) coordinates, the system implements an attribute-based naming schema.

In federated management architectures, the exchange of management data among neighboring nodes is confined to single, one-hop broadcast announcements. Further transmissions can be done in distance-vector fashion. If a rule or policy is required to be transmitted to every node of the network, the management system employs a simple dissemination protocol, e.g. a gossiping protocol, or a reliable protocol such as Directed Diffusion [Heidemann et al. 2001]. It is assumed that such transmissions in a federated system are sparse and thus the communication and energy overhead is minimized.

5. OPEN RESEARCH ISSUES

Research challenges for sensor network management systems are discussed from three perspectives; system design, distribution of management logic, and synergy with coexistent protocols. The aim is to establish a *lingua franca* among researchers, rather than presenting individual agenda items for each management area; their requirements have been summarized in Section 3.7.

Given management functional requirements, as perceived by the six areas of concern, we can obtain a number of solutions that deliver the required functionality by assembling different components together. Although functionally equivalent, every solution is associated with a different cost, in terms of resource consumption (e.g. energy, or bandwidth consumption) and non-functional guarantees (e.g. end-to-end delay, or mean time to repair). We wish to enable a user to specify the application non-functional requirements as objectives and constraints over specific system properties [Ma et al. 2005]. This way, a composite solution can be found superior to another functionally equivalent solution if it can better satisfy (i.e. optimize) those multiple objectives in a constrained environment. The DIAS project[4] encourages the realization of such a design methodology by constructing sensor systems that are heuristically optimal with respect to a global cost function.

The management architecture stems from the selected communication model for the collection and dissemination of management information. Despite the wireless multicast advantage, aggressive message broadcast may result in a broadcast storm; unnecessary transmissions may congest and thus degrade the performance of the network. In addition to complex routing protocols (e.g. directed diffusion [Heidemann et al. 2001]), minimization of the communication overhead introduced by the

[4]Design, Implementation and Adaptation of Sensor Networks through Multi-dimensional Co-design (DIAS-MC). For more information visit `www.dcs.gla.ac.uk/dias`.

management traffic to the network can also be addressed by design. For example, piggybacking of management information onto application, or other periodic, data packets can significantly reduce the communication overhead of the management system. Currently, piggybacking techniques have been used for flow control [Ee 2005] and aggregation of management data [Zhao et al. 2003].

Sensor network and system management has been tackled as a linear programming network optimization problem [Perillo and Heinzelman 2003]. In many cases, optimal network configurations are approximated in a centralized manner, by a global computational engine. Such approaches don't scale well and introduce additional latency to the system. Distributed optimization algorithms [Rabbat and Nowak 2004] can be exploited for the solution of the network management problem. Two issues must be considered. First, the problem formulation must be a joint optimization effort across all six management dimensions to capture the multiple objectives and constraints of the system. Second, the system must be able to infer universal knowledge, i.e. the current network status, at each optimization process locally.

A management system operates in service of the sensing application. Thus, management information must be available to coexistent application, networking, and operating system components. For example, routing protocols may utilize management data in order to specify optimal paths to sink node(s) [Hull et al. 2003; Younis et al. 2003]. Given an overall set of cost factors that characterize a link or a routing path (e.g. residual energy, number of hops, traffic load, packet success rate), the routing algorithm could optimize a routing decision according to the application requirements.

6. CONCLUSIONS

Management's functional requirements dictate *what* functionality the sensor system must provide to ensure correctness during its deployed lifetime. The system designer can then select *how* to address these requirements. Six functional dimensions (viz. deployment, diagnostics, application software, bandwidth, energy, and security management) enable a designer to navigate through the requirements' space. For each dimension, we have presented the state of the art in sensor management. Whereas management mechanisms exhibit similar functional behavior, they inevitably provide different service level guarantees; existing mechanisms are contrasted according to their data delivery model, communication model, system architecture, or resource requirements. The list of management components within each functional dimension is subject to continuous refinement; thus, our goal is to enable the designer to navigate through the management space and select (or create) mechanisms along the axes presented in this survey.

REFERENCES

Akkaya, K. and Younis, M. 2003. An energy-aware QoS routing protocol for wireless sensor networks. In *ICDCSW '03: Proceedings of the 23rd International Conference on Distributed Computing Systems*. IEEE Computer Society, Washington, DC, USA, 710.

Berman, P., Calinescu, G., Shah, C., and Zelikovsky, A. 2004. Power efficient monitoring management in sensor networks. In *IEEE Wireless Communications and Networking Conference*. Vol. 4. 2329–2334.

Capkun, S., Hamdi, M., and Hubaux, J. 2001. GPS-free positioning in mobile ad-hoc networks. In *HICSS '01: Proceedings of the 34th Annual Hawaii International Conference on System Sciences (HICSS-34)-Volume 9*. IEEE Computer Society, Washington, DC, USA, 9008.

Cerpa, A. and Estrin, D. 2004. ASCENT: adaptive self-configuring sensor networks topologies. *IEEE Transactions on Mobile Computing 3,* 3 (July-August), 272–285.

Chen, B., Jamieson, K., Balakrishnan, H., and Morris, R. 2001. SPAN: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. In *Proceedings of the Sixth Annual International Conference on Mobile Computing and Networking*. 85–96.

Deb, B., Bhatnagar, S., and Nath, B. 2003. Multi-resolution state retrieval in sensor networks. In *Proceedings of the First IEEE International Workshop on Sensor Network Protocols and Applications*. 19–29.

Ee, C. T. 2005. Argus: Bandwidth management in sensor networks. Tech. Rep. UCB/CSD-05-1373, EECS Department, University of California, Berkeley.

Eschenauer, L. and Gligor, V. D. 2002. A key-management scheme for distributed sensor networks. In *CCS '02: Proceedings of the 9th ACM conference on Computer and communications security*. ACM Press, New York, NY, USA, 41–47.

Frank, C. and Römer, K. 2005. Algorithms for generic role assignment in wireless sensor networks. In *SenSys '05: Proceedings of the 3rd international conference on Embedded networked sensor systems*. ACM Press, New York, NY, USA, 230–242.

Goldszmidt, G. and Yemini, Y. 1995. Distributed management by delegation. In *ICDCS '95: Proceedings of the 15th International Conference on Distributed Computing Systems*. IEEE Computer Society, Washington, DC, USA, 333.

He, T., Huang, C., Blum, B. M., Stankovic, J. A., and Abdelzaher, T. F. 2005. Range-free localization and its impact on large scale sensor networks. *Transactions on Embedded Computing Systems 4,* 4, 877–906.

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

Hsin, C. and Liu, M. 2002. A distributed monitoring mechanism for wireless sensor networks. In *WiSE '02: Proceedings of the 3rd ACM workshop on Wireless security*. ACM Press, New York, NY, USA, 57–66.

Huang, D., Mehta, M., Medhi, D., and Harn, L. 2004. Location-aware key management scheme for wireless sensor networks. In *SASN '04: Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks*. ACM Press, New York, NY, USA, 29–42.

Hui, J. W. and Culler, D. 2004. The dynamic behavior of a data dissemination protocol for network programming at scale. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*. ACM Press, New York, NY, USA, 81–94.

Hull, B., Jamieson, K., and Balakrishnan, H. 2003. Bandwidth management in wireless sensor networks. Tech. Rep. MIT-LCS-TR-909, Massachusetts Institute of Technology, Laboratory for Computer Science. April.

ISO. 1989. Information processing systems – open systems interconnection – basic reference model – part 4: Management framework. Tech. Rep. ISO/IEC 7498-4: 1989 (E), ISO/IEC. November.

Iyengar, R. and Sikdar, B. 2003. Scalable and distributed GPS free positioning for sensor networks. In *Proceedings of IEEE ICC*. 338–342.

Jaikaeo, C., Srisathapornphat, C., and Shen, C.-C. 2001. Diagnosis of sensor networks. In *ICC 2001: IEEE International Conference on Communications*. Vol. 5. 1627–1632.

Karlof, C., Sastry, N., and Wagner, D. 2004. TinySec: a link layer security architecture for wireless sensor networks. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*. ACM Press, New York, NY, USA, 162–175.

Kogekar, S., Neema, S., Eames, B., Koutsoukos, X., Ledeczi, A., and Maroti, M. 2004. Constraint-guided dynamic reconfiguration in sensor networks. In *IPSN'04: Proceedings of the third international symposium on Information processing in sensor networks*. ACM Press, New York, NY, USA, 379–387.

Levis, P. and Culler, D. 2002. Maté: 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*. ACM Press, New York, NY, USA, 85–95.

Levis, P., Patel, N., Shenker, S., and Culler, D. 2003. Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor networks. Tech. Rep. UCB/CSD-03-1290, EECS Department, University of California, Berkeley.

Liu, D. and Ning, P. 2003. Establishing pairwise keys in distributed sensor networks. In *CCS '03: Proceedings of the 10th ACM conference on Computer and communications security*. ACM Press, New York, NY, USA, 52–61.

Liu, T. and Martonosi, M. 2003. Impala: a middleware system for managing autonomic, parallel sensor systems. In *PPoPP '03: Proceedings of the ninth ACM SIGPLAN symposium on Principles and practice of parallel programming*. ACM Press, New York, NY, USA, 107–118.

Ma, H., Wang, D., Bastani, F. B., Yen, I.-L., and Cooper, K. 2005. A model and methodology for composition qos analysis of embedded systems. In *IEEE Real-Time and Embedded Technology and Applications Symposium*. 56–65.

Mainland, G., Parkes, D. C., and Welsh, M. 2005. Decentralized, adaptive resource allocationf for sensor networks. In *2nd Symposium on Networked Systems Design and Implementation (NSDI '05)*.

Marsh, D., Tynan, R., O'Kane, D., and O'Hare, G. M. P. 2004. Autonomic wireless sensor networks. *Engineering Applications of Artificial Intelligence 17,* 7 (October), 741–748.

Martinez, K., Hart, J. K., and Ong, R. 2004. Environmental sensor networks. *Computer 37,* 8, 50–56.

Moharrum, M. A. and Eltoweissy, M. 2005. A study of static versus dynamic keying schemes in sensor networks. In *PE-WASUN '05: Proceedings of the 2nd ACM international workshop on Performance evaluation of wireless ad hoc, sensor, and ubiquitous networks*. ACM Press, New York, NY, USA, 122–129.

Perillo, M. A. and Heinzelman, W. B. 2003. Sensor management policies to provide application QoS. *Ad Hoc Networks 1,* 2-3 (September), 235–246.

Rabbat, M. and Nowak, R. 2004. Distributed optimization in sensor networks. In *IPSN '04: Proceedings of the third international symposium on Information processing in sensor networks*. ACM Press, New York, NY, USA, 20–27.

Ramanathan, N., Chang, K., Kapur, R., Girod, L., Kohler, E., and Estrin, D. 2005. Sympathy for the sensor network debugger. In *SenSys '05: Proceedings of the 3rd international conference on Embedded networked sensor systems*. ACM Press, New York, NY, USA, 255–267.

Ruiz, L. B., Nogueira, J. M. S., and Loureiro, A. A. F. 2003. MANNA: a management architecture for wireless sensor networks. *Communications Magazine, IEEE 41,* 2 (February), 116–125.

Ruiz, L. B., Siqueira, I. G., e Oliveira, L. B., Wong, H. C., Nogueira, J. M. S., and Loureiro, A. A. F. 2004. Fault management in event-driven wireless sensor networks. In *MSWiM '04: Proceedings of the 7th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems*. ACM Press, New York, NY, USA, 149–156.

Savarese, C., Rabaey, J., and Beutel, J. 2001. Locationing in distributed ad hoc wireless sensor networks. In *Proc. 2001 Int'l Conf. Acoustics, Speech, and Signal Processing (ICASSP 2001)*. Vol. 4. IEEE, Piscataway, NJ, 2037–2040.

Schurgers, C., Tsiatsis, V., and Srivastava, M. B. 2002. STEM: Topology management for energy efficient sensor networks. In *IEEE Aerospace Conference Proceedings*. Vol. 3. 1099–1108.

Shen, C.-C., Srisathapornphat, C., and Jaikaeo, C. 2001. Sensor information networking architecture and applications. *IEEE Personel Communication Magazine 8,* 4 (August), 52–59.

Sinha, A. and Chandrakasan, A. 2001. Dynamic power management in wireless sensor networks. *IEEE Design and Test of Computers 18,* 2, 62–74.

Song, H. and Kim, D. 2005. UPnP-based sensor network management architecture. In *Second International Conference on Mobile Computing and Ubiquitous Networking*.

Stathopoulos, T., Heidemann, J., and Estrin, D. 2003. A remote code update mechanism for wireless sensor networks. Tech. Rep. CENS-TR-30, University of California, Los Angeles, Center for Embedded Networked Computing. November.

Su, W. and Lim, T. L. 2006. Cross-layer design and optimization forwireless sensor networks. *SNPD-SAWN 0*, 278–284.

Tolle, G. and Culler, D. 2005. Design of an application-cooperative management system for wireless sensor networks. In *Proceeedings of the Second European Workshop on Wireless Sensor Networks*. 121–132.

van Dam, T. and Langendoen, K. 2003. An adaptive energy-efficient MAC protocol for wireless sensor networks. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*. ACM Press, New York, NY, USA, 171–180.

Watro, R., Kong, D., fen Cuti, S., Gardiner, C., Lynn, C., and Kruus, P. 2004. TinyPK: securing sensor networks with public key technology. In *SASN '04: Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks*. ACM Press, New York, NY, USA, 59–64.

Xu, Y., Heidemann, J., and Estrin, D. 2001. Geography-informed energy conservation for ad hoc routing. In *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking*. ACM, Rome, Italy, 70–84.

Ye, W., Heidemann, J., and Estrin, D. 2002. An energy-efficient MAC protocol for wireless sensor networks. In *INFOCOM*.

Younis, M., Youssef, M., and Arisha, K. 2003. Energy-aware management for cluster-based sensor networks. *Computer Networks 43,* 5 (December), 649–668.

Zhao, Y. J., Govindan, R., and Estrin, D. 2003. Computing aggregates for monitoring wireless sensor networks. In *1st IEEE International Workshop on Sensor Network Protocols and Applications (SNPA 03)*. Anchorage, AK, USA, 139–148.

Zou, Y. and Chakrabarty, K. 2004. Sensor deployment and target localization in distributed sensor networks. *Trans. on Embedded Computing Sys. 3,* 1, 61–91.