Network Management in Wireless Sensor Networks

Winnie Louis Lee*, Amitava Datta*, and Rachel Cardell-Oliver*
School of Computer Science & Software Engineering
The University of Western Australia
35 Stirling Highway
Crawley, WA 6009, Australia
Phone:+61 8 6488 2305
Fax:+61 8 6488 1089

Contents

I. Introduction 1
A. Wireless Sensor Networks 2
B. Management Functionality 2
C. Chapter Overview 3

II. System Organisation 3
A. Management Reactivity 3
B. Management Architecture 3

III. Network Management System Design Criteria 5

IV. Network Management Systems 5
A. Sensor Network Management Framework 5
1. BOSS 5
2. MANNA 6
B. Sensor Network Management Protocols 8
1. RRP 8
2. SNMS 9
3. sNMP 10
4. WinMS 11
C. Management By Delegation 12
1. Agilla 12
2. Mobile Agent-Based Policy Management 12
3. Sectoral Sweeper 12
4. Intelligent Agent-Based Power Management 13
D. Debugging Tools 13
1. Sympathy 13
2. Two-Phase Self-Monitoring System 14
E. Visualisation Tool 15
1. MOTE-VIEW 15
F. Power Management Systems 15
1. SenOS 15
2. AppSleep 15
3. Node-Energy Level Management 16
G. Traffic Management Systems 16
1. Siphon 16
2. DSN Resource Management 17
H. Applications with Network Management Functions 17

V. Summary 18

Acknowledgments 18

References 18

*Electronic address: {winnie,datta,rachel}@csse.uwa.edu.au

I. INTRODUCTION

Network management is the process of managing, monitoring, and controlling the behaviour of a network. Wireless sensor networks (WSNs) pose unique challenges for network management that make traditional network management techniques impractical. In traditional networks the primary goals are minimizing response time and providing comprehensive information, but in sensor networks the primary goal is minimizing energy use [1] and the main means for doing this is by reducing the amount of communication between nodes. Optimizing the operational and functional properties of WSNs may require a unique solution for each application problem [2]. WSNs are highly dynamic and prone to faults, mainly because of energy shortages, connectivity interruptions, and environmental obstacles. Network failures are common events rather than exceptional ones [3]. Thus, in WSNs, we are mainly concerned with monitoring and controlling node communication in order to optimize the efficiency of the network, ensure the network operates properly, maintain the performance of the network, and control large numbers of nodes without human intervention.

A network management system designed for WSNs should provide a set of management functions that integrate configuration, operation, administration, security, and maintenance of all elements and services of a sensor network. We focus on applications that provide management schemes in terms of monitoring and controlling WSNs. Security management is beyond the scope of this chapter.

The main task of WSN monitoring is to collect information about the following parameters: node states (e.g., battery level and communication power), network topology, wireless bandwidth, link state, and the coverage and exposure bounds of WSNs. A sensor network management system can perform a variety of management control tasks based on the collected network states such as controlling sampling frequency, switching node on/off (power management), controlling wireless bandwidth usage (traffic management), and performing network reconfiguration in order to recover from node and communication faults (fault management).

Monitoring individual nodes in a large sensor network may be impractical. It is sufficient to control the network

by ensuring specific network coverage. Furthermore, sensor nodes are typically deployed in remote or harsh conditions and the configuration of nodes in WSNs changes dynamically. Thus, a sensor network management system should allow the network to self-forming, self-organise, and ideally to self-configure in the event of failures without prior knowledge of the network topology. Despite the importance of sensor network management, there is no existing generalized solution for WSN management [4]. However, most sensor network applications are designed with network management in mind and thus no extra network management layer is required. Readers familiar with wireless sensor networks may proceed to Section I B.

A. Wireless Sensor Networks

Wireless sensor networks are emerging applications of pervasive computing, consisting of many small, low-power, and intelligent sensor nodes (or motes) and one or more base stations. Sensor nodes gather information in diverse settings including natural ecosystems, battlefields, and man made environments [5–8] and send the information to one or more base stations. Sensor nodes work under severe resource constraints such as limited battery power, computing power, memory, wireless bandwidth, and communication capability, while the base station has more computational, energy and communication resources. The base station acts as a gateway between sensor nodes and the end user.

Sensor network applications use a data-centric approach that views a network as a distributed system consisting of many autonomously cooperating sensor nodes [9], any of which may have a role in routing, data gathering, or data processing. Every node will communicate through other nodes in a sensor network to produce information-rich results (e.g., temperature and soil-moisture in a certain region of the network). Furthermore, intermediate nodes can perform data aggregation and caching that is useful to reduce communication overheads [10, 11]. Sensor network applications can be categorized according to its operational paradigm: data gathering and event-driven. The data gathering application requires sensor nodes to periodically report their data to the base station. In the event-driven application, nodes only send data when an event of interest occurs.

B. Management Functionality

The function of network management systems is to monitor and control a network. These activities are wide ranging, and in this section we classify existing sensor network management systems in terms of the functionality they provide.

Systems for sensor networks that are based on **traditional network management systems** include BOSS [12] and MANNA [4]. BOSS serves as a mediator between UpnP networks and sensor nodes. MANNA provides a general framework for policy-based management of sensor networks. sNMP [13] provides network topology extraction algorithms for retrieving network state.

Other researchers have designed novel **routing Protocols** for network management. For example, TopDisc [3] and STREAM [14] are used in sNMP for extracting network topology, RRP [15] uses a zone-flooding protocol, SNMS [16] introduces the Drip protocol, and WinMS [17] is based on the FlexiMAC protocol.

Fault detection is an important focus of the systems TP [18], Sympathy [19], MANNA [20], and WinMS [17]. In TP, each node monitors its own health and its neighbours' health, so providing local fault detection. Sympathy goes one step further by providing a debugging technique to detect and localize faults that may occur from interactions between multiple nodes. MANNA performs centralised fault detection based on analysis of gathered WSN data. In WinMS, there is a scheduled period where nodes listen to their environment activities and can self-configure themselves in the event of failure without prior knowledge of the full network topology. In addition, WinMS provides a centralised fault management scheme that analyses network states to detect and to predict potential failures and takes corrective and preventive actions accordingly. TP, Sympathy, and MANNA focus solely on fault detection and debugging, they provide no automatic network reconfiguration to allow the network to recover from faults and failures. WinMS differs from the rest as it adaptively adjusts the network by providing local and central recovery mechanisms.

TinyDB [21] and MOTE-VIEW [22] are **visualisation tools** that provide graphical representations of network states to the end user. TinyDB is a query-based interface that allows the end user to retrieve information from sensor nodes in a network. MOTE-VIEW also allows the end user to control sensor node settings such as transmission power, radio frequency, and sampling frequency. In these systems the central server analyses data collected from the network. The main disadvantage of such passive monitoring schemes is that they are not adaptive to current network conditions, and provides no self-configuration in the event of faults.The end user must manually manage the network and interpret the graphical representation of collected data.

Several network management systems focus on **managing power resources** in the network: Agent-Based Power Management [23], SenOS [9], AppSleep [24], and Node-Energy Level Management [25]. Agent-Based Power Management utilizes intelligent mobile agents to manage sub-networks and perform local power management processing. It can reduce the sampling rate of nodes with critical battery and reduce node transmission power. Other systems such as SenOS, AppSleep, and Node-Energy Level Management [25] use common sensor nodes to perform power management. SenOS and AppSleep put nodes to sleep when they are not needed.

Node-Energy Level Management [25] allows nodes to reject a management task based on the importance of that task.

Traffic management functions are provided in Siphon [26], DSN RM [27] and WinMS [17] . Siphon [26] uses multi-radio nodes to redirect traffic from common-nodes in a network in order to prevent congestion at the central server and in the primary radio network. In contrast, DSN RM [27] uses single-radio common-nodes to evaluate each of their incoming and outgoing links and apply delay schemes to these links when necessary in order to reduce the amount of traffic in the network. Congestion can also be avoided by modifying sensor nodes' reporting rate based on the reliability level of sensor node data [28]. By reducing a node's reporting rate, the number of packets transmitted in the network is reduced, so avoiding potential congestion. WinMS [17] can reconfigure nodes to report their data more rapidly or slowly depending on the significance and importance of their data to the end-user. The WinMS scheme supports non-uniform and reactive sensing in different parts of a network.

Several network management protocols reviewed are **application specific** rather than general purpose schemes. RRP [15] is tailored for real-time data gathering applications with bursty and bulky traffic. AppSleep is designed for latency-tolerant applications. TP is a fault detection system best suited for monitoring and surveillance applications. SenOS power management is designed for SenOS operating system-based sensor networks. Network management systems such as RRP and Siphon [26] require special hardware: RRP uses GPS nodes for implementing the proposed zone flooding protocol, while Siphon requires multi-radio nodes to act as virtual sinks.

C. Chapter Overview

The remainder of this chapter discusses different approaches to network management system organisation. Section III identifies criteria that must be satisfied by network management systems for sensor networks. Section IV reviews state of the art network management systems for sensor networks. Finally, Section V provides open research issues on sensor network management design.

II. SYSTEM ORGANISATION

There are two main choices for the system organisation of sensor network management protocols: central vs distributed control, and reactive vs proactive monitoring. Table I summarizes state of the art sensor network management systems according to these system choices.

A. Management Reactivity

Sensor network management systems can be classified according to the approach taken to monitoring and control.:

- **Passive monitoring.** The system collects information about network states. It may perform post-mortem analysis of data.
- **Fault detection monitoring.** The system collects information about network states in order to identify whether faults have occurred.
- **Reactive monitoring.** The system collects information about network states to detect whether events of interest have occurred and then adaptively reconfigure the network.
- **Proactive monitoring.** The system actively collects and analyses network states to detect past events and to predict future events in order to maintain the performance of the network.

B. Management Architecture

Sensor network management systems can also be classified according to their network architecture [7]: centralised, distributed, or hierarchical.

In **centralised management systems**, such as BOSS [12], MOTE-VIEW [22], SNMS [16], and Sympathy [19], the base station acts as the manager station that collects information from all nodes and controls the entire network. The central manager with unlimited resources can perform complex management tasks, reducing the processing burden on resource-constrained nodes in the sensor network. Since the central manager also has the global knowledge of the network, it can provide accurate management decisions. But, this approach has some problems. First, it incurs a high message overhead (bandwidth and energy) from data polling, and this limits its scalability. Second, the central server is a single point of data traffic concentration and potential failure. Lastly, if a network is partitioned, then nodes that are unable to reach the central server are left without any management functionality.

Distributed management systems employ multiple manager stations. Each manager controls a subnetwork and may communicate directly with other manager stations in a cooperative fashion in order to perform management functions. Distributed management has lower communication costs than centralised management, and so provides better reliability and energy-efficiency. But it is complex and difficult to manage. Distributed management algorithms may be computationally too expensive for resource-constrained sensor network nodes. Distributed management systems include DSN RM [27], Node-energy level management [25], AppSleep [24], and sensor management optimization [28].

TABLE I: Network management system organisation.

Network Management System	Reactivity	Architecture
WinMS	Proactive	Hierarchical
DSN RM	Proactive	Hierarchical
Mobile agent-based policy management	Proactive	Hierarchical
Intelligent agent-based power management	Proactive	Distributed
Siphon	Proactive	Distributed
BOSS	Proactive	Centralised
Sympathy	Proactive	Centralised
SenOS	Reactive	Hierarchical
Agilla	Reactive	Distributed
Node-energy level management	Reactive	Distributed
Two-phase monitoring system	Fault-detection	Distributed
TopDisc	Passive	Hierarchical
AppSleep	Passive	Hierarchical
RRP	Passive	Hierarchical
STREAM	Passive	Hierarchical
Sectoral Sweeper	Passive	Distributed
TinyDB	Passive	Centralised
MOTE-VIEW	Passive	Centralised
SNMS	Passive	Centralised
MANNA	N/A	N/A

Another disadvantage of distributed systems is memory costs. For example, neighborhood state transition diagram maintenance in TP [18], task usage profile maintenance in Node-Energy Level Management [25], and tuple space maintenance in Agilla [29] all require significant memory resources.

A mobile agent-based framework is an example of distributed management system implementation. Network management systems that use this approach are Agilla, Sectoral Sweeper [30], Mobile Agent-Based Policy Management [31], Agent-Based Power Management [23], and MANNA [4]. The main advantages of these approaches are that local processing reduces network bandwidth requirements and prevents network bottlenecks by reducing processing at the central server [31]. Furthermore, agents can be designed to distribute tasks in the network. For example, agents can relay some tasks from overloaded nodes to other nodes with lower workloads. In addition, agents can be moved flexibly to cover an area of interest [4] and agents can shift debugging and transmission operation from low-power sensor nodes to extend network lifetime.

There are several drawbacks of agent-based approaches. First, there is a need for special nodes to perform management tasks. Second, the human manager needs to locate these agents 'intelligently' in order to cover all nodes in the network. Thus, this approach requires a network to be configured manually and the human manager needs to have expertise about the optimal number of agents as well as agent location for a particular sensor network application. Third, the agent-based approach introduces delays when a manager wants to retrieve network states of a node because the manager needs to wait for an agent to visit the node [14]. Fourth, the agent-based approach does not scale for large WSNs because as the number of sensor nodes increases, the number of agents deployed must be increased. Alternatively, reducing the number of agents increases the time required for an agent to visit nodes in a network. Finally, since the agent typically sends aggregated management information from a set of managed nodes in a network, fine-grained information from individual nodes is compromised.

Hierarchical network management is a hybrid between the centralised and distributed approach. Intermediate managers are used to distribute management functions, but do not communicate with each other directly. Each manager is responsible for managing the nodes in its sub-network. It passes information from its sub-network to its higher-level manager, and also disseminates management functions received from the higher-level manager to its sub-network. For example, in AppSleep [24], TopDisc [3], STREAM [14], and SenOS [9], some common-nodes are selectively elected as cluster heads to act as distributed managers. There is a non-trivial energy overhead for selecting cluster heads. Agent-based policy management [31] uses mobile agents as distributed managers. In RRP [15], individual nodes have distinct roles: either acquiring raw sensor data, transporting data, or filtering data. Unlike other systems, DSN RM [27] and WinMS [17] allow individual nodes to act as agents and perform management functions autonomously based on their neighborhood states.

III. NETWORK MANAGEMENT SYSTEM DESIGN CRITERIA

A network management system designed for WSNs must take into account the unique properties of WSNs. The following criteria are used to evaluate the sensor network management systems reviewed in this chapter:

1. **Lightweight operation**. A system should be able to run on sensor nodes without consuming too much energy or interfering with the operation of the sensor nodes. Lightweight operation prolongs network lifetime.

2. **Robustness and fault tolerance**. WSNs are prone to network dynamics such as dropped packets, nodes dying, becoming disconnected, powering on or off, and new nodes joining the network. A management system should be resilient to network dynamics by reconfiguring the network as required.

3. **Adaptability and responsiveness**. A system should be able to retrieve and adapt to the current network states or changing network conditions including changes in network topology, node energy level, and the coverage and exposure bounds of WSNs.

4. **Minimal data storage**. A data model used to represent management data must be extensible and able to accommodate information needed to perform the management functions, but must also respect the memory constraints of WSNs.

5. **Scalability**. A system should operate efficiently in any network size.

Table II compares sensor network management systems according to these network management criteria.

IV. NETWORK MANAGEMENT SYSTEMS

Having identified different functionalities supported in existing network management systems, choices for system operation, and criteria for effective operation in a sensor network, we now examine individual state of the art sensor network management systems in more detail.

A. Sensor Network Management Framework

1. BOSS

Song et al. [12] propose a service discovery management architecture for WSNs. The architecture is based on UPnP, the standard service discovery protocol for network management. However, UPnP only runs on devices with high computation power and large memory. Thus, resource-constrained sensor nodes are unable to process the UPnP protocol. Song et al. address this issue by implementing an UPnP agent in the base station, called Bridge Of the SensorS (BOSS), which provides a bridge between a managed sensor network and a UPnP network. The proposed system consists of three main components: UPnP control point, BOSS, and non-UPnP sensor nodes. The control point is a powerful logical device with sufficient resources to run the UPnP protocol and manage a sensor network using the services provided by BOSS, e.g. PCs, PDAs, and notebooks. BOSS is a base node that acts as the mediator between non-UPnP sensor nodes and UPnP control point and is implemented in the base station. Each node in a sensor network is a non-UPnP device with limited resources and sensing capability.

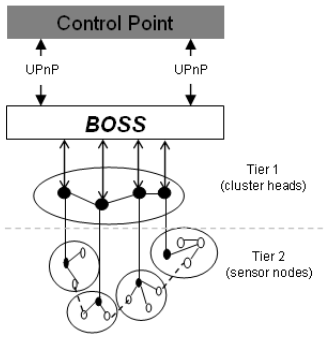

FIG. 1: BOSS Architecture. Reprinted with permission from [12].

FIG. 1 is an abstraction of the BOSS architecture detailed in [12]. The control point and BOSS use UPnP protocol to communicate with each other, while non-UPnP sensor nodes and BOSS use a sensor network proprietary protocol for communication. Thus, the human manager can implement any sensor network protocol to transport data from non-UPnP sensor nodes to the base node. BOSS has three functions. First, it is used to transfer UPnP messages between the sensor network and the control point. Second, since UPnP protocol uses XML message format that is different from a sensor network specific network message format, BOSS is responsible for interpreting transferred UPnP messages. Lastly, BOSS gathers network management information from sensor nodes to provide network management services from each sensor node to the control point. The base node stores network management services and performs all network management processing. Thus, the base node carries the network management computation burden, rather than the resource-constrained sensor nodes. In the BOSS architecture, the control point can specify which events of non-UPnP sensors it is interested in. BOSS then processes events reported by non-UPnP sensors to the control point.

The network management services provided by the BOSS include basic network information, localization, synchronization, and power management. BOSS can re-

TABLE II: Network management system evaluated against design criteria.

Network management system	Main management functionalities	Energy efficiency	Robustness	Adaptability	Memory efficiency	Scalability
WinMS	Network state retrieval, synchronization, local repair, and systematic resource transfer	Yes	Yes	Yes	Yes	Yes
BOSS	Network state retrieval, localization, synchronization, and power management	Yes	Yes	Yes	Yes	No
Mobile Agent-Based Policy Management	Policy-based management framework	Yes	Yes	Yes	Yes	No
AppSleep	Power management (extended sleeping schedules)	Yes	Yes	No	Yes	Yes
Intelligent Agent-Based Power Management	Local power management and sampling frequency control	Yes	No	Yes	Yes	No
Sympathy	Fault debugging	Yes	Yes	Yes	Yes	No
Two-Phase Monitoring System	Local fault detection schemes	Yes	Yes	Yes	No	Yes
SNMS	Query-based network health data collection and event logging	Yes	Yes	No	Yes	No
MOTE-VIEW	Network state visualisation and network state retrieval	Yes	No	No	Yes	Yes
Agilla	Event detection	Yes	No	Yes	Yes	No
STREAM	Network topology extraction	Yes	No	No	Yes	Yes
TopDisc	Network state retrieval (e.g. network topology and node energy level)	Yes	No	No	Yes	Yes
RRP	Data aggregation and zone flooding scheme	Yes	No	No	Yes	No
Sectoral Sweeper	Switching node on/off	Yes	No	No	Yes	No
Node-Level Management	Power management (task rejection)	Yes	No	No	Yes	No
SenOS	Switching node on/off	Yes	No	No	Yes	No
DSN RM	Priority-based traffic management and congestion avoidance scheme	Yes	No	Yes	No	No
Siphon	Multi-radio on-demand traffic management	No	No	Yes	Yes	No
MANNA	Policy-based management framework, network state retrieval, sampling frequency control, coverage maintenance, and fault detection	NA	NA	NA	NA	NA

trieve basic network state information from the sensor network, including sensor node device description, the number of sensor nodes in the network, and the network topology. The localization service gives location information for each sensor in the network. The synchronization service is responsible for performing clock synchronization among nodes in the network. For example, when a node is added in the network, the node sends a message to BOSS and BOSS informs the control point. The central point then synchronizes the new node with other nodes in the network. The power management service allows a human manager to manage the power of the sensor nodes by checking remaining battery and changing the operation mode of sensors.

The advantage of using BOSS is that different sensor network applications (e.g. structural monitoring, fire detection, and auto light control) can be managed by multiple UPnP control points (e.g. PCs and PDAs). Furthermore, BOSS allows a sensor network to adapt to topology changes and so supports proactive network management. A drawback of BOSS is that it requires an end-user to observe network states and take management actions accordingly.

2. MANNA

MANNA (a Management Architecture for Wireless Sensor Networks [4]), is a policy-based management system that collects dynamic management informa-

tion,maps this into WSN models, and executes management functions and services based on WSN models. MANNA's management policy specifies management functions that should be executed if certain network conditions are met. WSN models maintain the information about the state of the network. MANNA defines the relationship among WSN models in a Management Information Base (MIB). Some examples of WSN models include:

- **Topology map** depicting node connectivity and reachability of the network.
- **Residual energy map** showing battery level of nodes in the network.
- **Sensing coverage area map** describing the area covered by sensor elements.
- **Communication coverage area map** presenting communication range of nodes in a network.
- **Audit map** describing the security status of sensor nodes in a network, whether nodes have been attacked.

In MANNA, a management service consists of one or more management functions. Some of the management functions are coverage area supervision, network operating parameter configuration, topology map discovery, network connectivity discovery, aggregation discovery, energy map generation, and node localization discovery. The execution of management services depends on the information obtained from the WSN models. For example, the human manager can perform coverage area management service based on the information obtained from the energy map and the network topology map in order to determine unmonitored areas in a network.

MANNA adapts to dynamic WSN behaviours by analysing and updating the MIB. MIB update is a centralised operation and expensive in terms of energy consumption. Moreover, WSN uncertainties and delay may affect the accuracy of collected management information. To keep the MIB up-to-date, it is critical to determine the right time to query for management information and the right frequency for obtaining management information. In [4], Ruiz et al. focus solely on designing a network management architecture. They propose no specific MAC or routing protocols. However, they suggest the use of an agent-based framework to distribute management functions to managed systems in which agents collect management information from sensor nodes and transport these information back to the base station.

MANNA network management protocol (MNMP), is a lightweight protocol for managing information exchange among management entities (cluster heads, common-nodes, and manager) [32]. Basically, sensor nodes are organised in clusters (sub-network) and send their states to the agent located in the cluster-head. MNMP places management agents on the cluster-heads and each cluster-head acts as a manager for a cluster (local manager). Cluster-heads are responsible for executing local management functions and they aggregate management data received from sensor nodes. Cluster heads forward management data directly to the base station. Furthermore, cluster-heads can work cooperatively with other cluster-heads to achieve an overall management goal, for example, forming groups of nodes (i.e., clusters). A manager is a powerful management entity located outside the WSN responsible for complex management tasks requiring global knowledge of the network. This approach achieves energy efficiency and increases the accuracy of management decisions. Subramanian et al. [33] propose a similar hierarchical architecture approach for monitoring sensor networks, in which nodes report updates to their cluster heads periodically. However, the efficiency of this hierarchical monitoring approach depends on the size of the cluster.

Fault management for event-driven applications of WSNs can be performed using automatic management services provided by MANNA [20]. Fault management aims to detect failures in the network by analysing WSN models. The system provides two main management services: coverage area maintenance service and failure detection service. The central manager uses the topology map model and the energy model to build a coverage area model in order to monitor areas of sensing and communication coverage. The central manager can command the agent to execute a failure detection management service. For example, the manager sends MNMP GET requests for node state retrieval and uses GET-RESPONSEs to build the WSN audit map. If an agent or a node does not answer a GET request, the manager consults the energy map to verify whether the node has residual energy. If the node is still alive, the manager notifies a GET-RESPONSE failure to the end user. This scheme has a drawback of possibly providing false debugging diagnostics. For instance, common-nodes may be disconnected from their cluster-head and so unable to receive the GET operation from the manager. Or, GET and GET-RESPONSES packets may be lost as a result of noise in the environment. Random distribution and limited transmission range capability of common-nodes and cluster-heads provide no guarantee that every common-node can be connected to a cluster head. The proposed fault management strategy relies on the manager to perform system debugging. The scheme introduces no extra management processing overhead and runs independently of network failures. There are, however, still transmission costs incurred for network state polling. This management cost can be justified because it allows the manager to be aware of changing network conditions and to act upon them accordingly. The proposed system is a fault-detection monitoring system only and so the network is not reconfigured in the event of failures. Staddon et al.[34] proposes a similar centralised management approach, whereby the manager monitors the health of individual sensor nodes to detect node fail-

ures in a network. Unlike MANNA [20], it provides a method for recovering corrupted routes.

In [35], Ruiz et al. analysed different network configurations (homogenous or heterogenous and flat or hierarchical) and evaluated their performance. In a homogenous network, all nodes have the same hardware capabilities (battery, memory, processor, and communication device). In a heterogeneous network, these capabilities differ between nodes. In a heterogeneous hierarchical network, nodes with higher capabilities become cluster heads throughout the network lifetime, while a homogeneous network requires nodes to alternate the role of cluster heads. In a sensor network that implements the MANNA management scheme, nodes report their state (e.g. location and battery level) in addition to their sensing data. The manager maps the received data into topology, energy and coverage maps and performs management services to maintain the performance of the network. In homogenous flat WSNs, each node individually sends its states using an MNMP SENSOR-REPORT message to the base station. In homogenous or heterogeneous hierarchical WSNs, cluster-heads receive management data from their cluster nodes, perform necessary data processing, and send the processed data to the base station.

The MANNA framework can also be used for decentralised management using policies that define the desired self-service behavior of each sensor node according to local information [36] so that networks can manage themselves without direct human intervention. The decentralised management system offers 1) an agent-based service negotiation to allow the network to be more reactive to changing network conditions and 2) self configuration schemes to save energy, control network density, and maintain network coverage. The system implements service negotiation by pre-programming common nodes to adjust their services (sensing, processing, and data dissemination rate) based on their states and local environment conditions, and using cluster-heads to act as agents that monitor, analyse node message priority, and filter data sent by common nodes in a cluster. For example, in a fire-tracking application, common nodes sense temperature in their environment and determine the fire risk based on temperature data values. These nodes then assign a priority value to their packets (high or low depending on the fire risk) and transmit the packets to their respective cluster-head. Cluster heads evaluate the priority status of all received packets. If cluster heads receive a high-priority packet, they will discard low priority packets and start to aggregate high-priority packets for a period of time, before sending the aggregated data to the base station. Thus, local event detection approach allows delivery of data with different priorities. Event-detection results in an increase in network traffic, but it allows the network to be highly reactive since the manager is notified of an event earlier and with more accuracy than without this scheme. The system also provides self-management and self-configuration by allowing nodes to change their parameter values dynamically in response to changing network conditions. Cluster heads can change their transmission power by reconfiguring their communication range according to their distance from the base station and common nodes in their clusters. Moreover, the system utilizes DPM [37] to perform coverage area maintenance (network density control) by turning off redundant nodes in a network, and hence reducing congestion, collision, and energy waste.

B. Sensor Network Management Protocols

1. RRP

Liu et al. [15] propose a hybrid data dissemination framework, RRP, based on supply chain concept for managing data gathering applications such as habitat monitoring and battlefield surveillance. In business, supply chain management is a coordinated system of entities and activities for delivering a product or service from supplier to customer. The primary objective of supply chain management is to fulfill customer demands through the most efficient use of resources by allowing entities such as manufacturer, distributors, and retail outlets to have their own inner activities and management strategies, but to work cooperatively to achieve a common management goal. Liu et al. [15] apply the supply chain concept in the design of RRP by partitioning a sensor network into several functional regions, applying different routing schemes to different regions, and designing cooperation among different regions to provide better network performance in terms of reliability and energy usage. Liu et al. [15] also propose a novel zone flooding scheme that is a combination of flooding and geometric routing techniques.

RRP is a hierarchical system consisting of three areas: manufacturing area, transportation area, and warehouse and service area. It manages the acquisition of raw data from the manufacturing area to the delivery of processed data to the warehouse and service area. In RRP, sensor nodes are heterogenous in that they have different roles and tasks. In the manufacture area, sensor nodes are either source nodes that generate raw data or aggregation nodes responsible for filtering raw data. The aggregation node selects a transportation method and the flooding zone for forwarding the filtered data before putting the data into the transportation area. The transportation method can be either single-zone flooding or multi-zone flooding. The aggregation node uses a single flooding zone when the network traffic is low. For large data loads, the aggregation node fragments the data into several packets and uses multiple flooding zones to deliver the packets to multiple base stations in different flooding zones.

In the transportation area, sensor nodes collaboratively relay received data from the manufacture area to one or more base stations. RRP uses a zone-flooding

scheme to reduce the cost of topology maintenance and route discovery. The basic idea is to allow a node receiving a packet to determine a flooding-zone (bounded by two ellipses), based on parameters specified in the packet. The node then decides whether it is inside or outside the flooding zone by evaluating its location against the received zone parameters. Finally, the node rebroadcasts the packet if it is located within the flooding zone. This forwarding-decision scheme allows nodes in the manufacture area to manage energy distribution among nodes in the transportation area by selecting parameters. The main advantage of this scheme is that forwarding decisions are made locally in the network. However, the aggregation node in the manufacture area has the computational burden of determining the right parameters for bounding a flooding zone. It should ensure that a flooding zone has sufficient nodes to forward packets while maintaining high energy efficiency.

Sensor nodes in the warehouse and service areas are responsible for managing or reducing information implosion at base stations. In this area, RRP uses a modified SPIN [38] protocol instead of zone flooding as the underlying routing protocol. SPIN allows nodes in a neighborhood to communicate with each other and use meta-data negotiation (ADV-REQ-DATA) to eliminate the transmission of redundant data. RRP executes the ADV-REQ-DATA exchange through unicasting between a warehouse node and a base station that are maybe several hops away from each other. For example, packets that report the same event for base station *B1* are received by warehouse *W1* and warehouse *W2* located in the same flooding zone. Warehouse *W1* and *W2* then send ADV messages to *B1* and *B1* decides which warehouse can send data to it based on pre-programmed criteria such as hop distance or delay. If *B1* chooses *W1*, it sends a REQ to *W1*. Then, *W1* unicasts the requested data via a DATA message to *B1*. In this way, redundant packets can be eliminated, so reducing energy consumption.

The main advantages of RRP are that zone flooding ensures low message overheads, and adjusting the size of flooding zone ensures high reliability. RRP allows the end user to predefine the size of warehouses and flooding zones in order to achieve desired energy consumption, end-to-end delay, and routing overhead, while maintaining high packet delivery (reliability). Generally, the larger a flooding-zone, the more transportation nodes are involved in the packet forwarding, giving higher reliability but higher delay [15]. The drawbacks of RRP are that it requires GPS-attached nodes in order to implement the zone-flooding protocol and it requires a human manager to place sensor nodes in the field strategically at the initial network setup in order to support RRP hierarchical network management. The human manager has to decide which and where nodes should be located for the manufacture, transportation, and warehouse and service areas.

2. SNMS

Tolle and Culler [16] propose SNMS, a Sensor Network Management System. SNMS is an interactive system for monitoring the health of sensor networks. SNMS provides two main management functions: query-based network health data collection and event logging. The query system allows the user to collect and monitor physical parameters of the node environment. For example, the value of a node's remaining battery power can be used to predict node failures. Furthermore, temperature and humidity surrounding the sensor node can be indicators of upcoming failure. The event-driven logging system allows the user to set event parameters and nodes in the network will report their data if they meet the specified event thresholds.

SNMS supports two traffic patterns: Collection and Dissemination. Collection is used to obtain health data from the network and dissemination is used to distribute management messages, commands, and queries. SNMS uses a data gathering tree to collect network health information from sensor nodes in the network. The SNMS collection tree construction protocol uses flooding with random staggering of retransmission times for each node. In SNMS, every node in the network only maintains the single best parent based on the strongest received signal strength. The advantage of this approach is that it minimizes memory usage by not requiring sensor nodes to maintain a neighborhood table. Furthermore, SNMS minimizes traffic by only constructing a tree in response to messages sent from the base station [16]. The SNMS tree construction protocol is also adaptive to changing network condition since nodes will select a new parent if their existing parent dies. In SNMS, network states such as a node's current parent and link quality are only updated based on user queries.

SNMS proposes the Drip protocol to reliably disseminate messages, command, and queries to a set of managed nodes in the network. When a component (a user or sensor nodes) wants to make a query, it selects a specific identifier that represents a reliable delivery channel. The Drip protocol then transports messages or replies received on that channel to the component by requiring every sensor node to regularly check the channel it subscribes to, cache and extract data from the latest message received on that channel, and return a reply. The advantage of this dissemination approach is that it is application independent and thus it is able to provide management functions even when the application fails. However, when a component queries several independent variables, the Drip protocol needs to make a trade off between channel usage and node caching. Tolle and Culler [16] propose two methods to address this issue. The first method is to force components to collect the current value of each variable into a single message. This approach can ensure independent reliability of every variable only if each node records the same value for each variable. The second method is to attach a unique key for each variable

instead of reserving a channel for each message. However, this approach requires a large key space, and so high memory usage.

The main advantage of SNMS is that it introduces overhead only for human queries and so has minimal impact on memory and network traffic. SNMS further minimizes energy consumption by bundling the results of multiple queries into a single message instead of returning results individually. The main drawbacks of SNMS are that the network management function is limited to passive monitoring only, requiring human managers to submit queries and perform post-mortem analysis of management data. Furthermore, SNMP's centralised-processing approach requires continuous polling of network health data from managed nodes to the base station, and this can burden sensor nodes that should minimize transmissions in order to extend network lifetime.

3. sNMP

Deb et al. [13] propose a management framework called Sensor Network Management Protocol (sNMP). The sNMP framework has two functions. First, it defines sensor models that represent the current state of the network and defines various network management functions. Second, it provides algorithms and tools for retrieving network state through the execution of the network management functions. Models for sensors include network topology (node connectivity), energy map (node battery power), and usage patterns [3]. The correlation between the energy map and network topology can be used to identify weak areas in the network. Usage patterns describe network activity such as node duty cycles and bandwidth utilization in terms of amount of data transmitted per unit of time. Deb et al. [3] suggest that sensor models could be used for different network management functions. The human manager could use the current knowledge of network topology for future node deployment. By measuring network states periodically, the human manager can monitor and maintain the network by identifying which parts of the network have a low performance, and taking corrective actions as necessary. From periodic monitoring of network states, the human manager could also analyse network dynamics to predict network failures and then take preventive actions.

In the sNMP framework, sensor models form the Management Information Base (MIB) for sensor networks [13]. Network topology depicts states of the network that are useful for determining the number of active nodes and the connectivity of nodes in the network [3], [14]. Deb et al. propose a topology discovery algorithm, TopDisc [3] and Sensor Topology Retrieval at Multiple Resolutions, STREAM, [14] for retrieving network topology.

The TopDisc algorithm provides a clustering mechanism that allows a minimal set of nodes to be active in maintaining network connectivity. The algorithm selects a set of distinguished nodes and forms a network topology based on the nodes' neighborhood information. Nodes listen to other nodes within their communication range to collect local neighborhood information. Management is done by exchanging local neighborhood information between adjacent clusters.

TopDisc has three management functions: network state retrieval, data dissemination and aggregation, and duty cycle assignment. The cluster heads are used to retrieve network states such as network topology map, energy map, and usage pattern. A Tree of clusters (TreC) consists of a number of sensor nodes that are optimal in the number of hops from the cluster head monitoring node. This approach allows efficient data dissemination and data aggregation. Management information routing is performed by inter-cluster communication. This routing approach allows the human manager to specify rules for forwarding in such a way that loads can be distributed among nodes and fair duty cycle assignment is ensured. For example, a node may refuse to be a forwarding node if it has insufficient energy. An advantage of the TopDisc approach is that it provides a framework to perform network management functions based on local information that is highly scalable. However, clustering introduces overheads of cluster-head election and cluster maintenance that are expensive in terms of latency and energy.

Retrieving a complete network topology of large sensor networks is an expensive operation for resource-constrained sensor nodes. Two common methods for retrieving network states, direct response and aggregated response, incur a large communication overhead [14]. The overhead is proportional to the number of hops a node's reply has to travel to reach the base station. To address this issue, Deb et al. [14] propose a distributed, parameterized algorithm, STREAM, which is designed to return network topology at a required Resolution, at proportionate costs. STREAM makes a trade-off between topology details and resources usage [14].

STREAM proposes a *Minimal Virtual Independent Dominating Set* (MVDIS) mechanism in which a minimal set of nodes are selected to extract network topology at a desired resolution. Thus, only a subset of network nodes reply to topology discovery queries with their neighborhood information. STREAM uses a coloring algorithm to create a MVIDS(r) based on the virtual range r. STREAM selects the nodes in MIVDS(r) to reply to topology discovery queries. Distance r is the topology resolution control parameter. The creation of a MVIDS only incurs message complexity of N (number of nodes in the network) and hence STREAM is highly energy efficient, in comparison with a centralised scheme. The selection of a representative set of nodes in a sensor network is important to give an accurate picture of network states. Since the problem of finding a minimum dominating set is NP-complete, STREAM uses a heuristic greedy approach for determining the set of dominating nodes [14]. The main advantage of STREAM is that the hu-

man manager can flexibly control the topology resolution and the resource expended. A drawback of STREAM is the significant computation overhead of finding and maintaining an optimum set of dominating nodes.

4. WinMS

Louis Lee et al. [17] propose an adaptive policy-based management system for WSNs, called Wireless Sensor Network Management System (WinMS). The end user predefines management parameter thresholds on sensor nodes that are used as event triggers, and specifies management tasks to be executed when the events occur. WinMS adapts to changing network conditions by allowing the network to reconfigure itself according to current events as well as predicting future events, in order to maintain the performance of the network and achieve effective networked node operations. FIG. 2 shows the WinMS architecture [17]. FlexiMAC [39] is the underlying MAC and routing protocol that schedules node communication and continuously and efficiently collects and disseminates data, to and from sensor nodes in a data gathering tree. A local network management scheme provides autonomy to individual sensor nodes to perform management functions according to their neighborhood network state, such as topology changes and event detections. The central network management scheme uses the central manager with a global knowledge of the network to execute corrective and preventive management maintenance. The central manager maintains an MIB that stores WSN models that represent network states. The central manager analyses the correlation among WSN models to detect interesting events such as areas of weak network health, possible network partition, noisy areas, and areas of rapid data changes.

FlexiMAC is a TDMA-based protocol that provides synchronized communication using a loose slot structure. Initially, all nodes in a network build their data gathering schedules. After this initial one-off, global setup phase, all repair operations are local. Nodes can claim or remove a time slot based on current information in their lookup table without exchanging information with any other nodes in the network prior to modifying their schedules. Thus, it is not necessary to specify the number of slots required for the network in advance. In FlexiMAC, networked nodes sleep when they are not scheduled to transmit, receive, or listen. FlexiMAC also has a scheduled short CSMA period in every data gathering cycle, where all nodes in the network listen for distress signals sent by 'faulty' nodes in the environment. For example, when a new node is added to the network or an existing node in the network wants to find a new parent, FlexiMAC allows nodes in the network to adopt these nodes as their children, assign schedules for them, and rebuild their own schedules. FlexiMAC's flexible slot structure makes it strongly fault-tolerant and also highly energy-efficient. WinMS utilizes FlexiMAC's scheduling

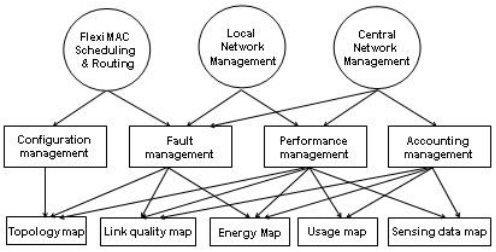

FIG. 2: WinMS architecture. Reprinted with permission from [17].

approach to piggyback management data on data packets and hence eliminating energy and bandwidth overheads of regular network state update.

In WSNs, a small part of a network may need to increase the rate of data gathering significantly in order to report important data in real-time. For example, it is critical that sensor nodes report their data more often when they detect event triggers such as sensor readings changing rapidly or exceeding user-specified thresholds. WinMS addresses this issue by a systematic resource transfer function that allows resources (time slots) from one part of the network to be transferred to another part of the network. WinMS uses FlexiMAC to support resource transfer among nodes in the network and to ensure slots are systematically transferred among nodes, maintaining low energy overhead and fast convergence. Using FlexiMAC, the network can be reconfigured and stabilized within a single data gathering cycle. The systematic resource transfer function can be executed locally or centrally. In the decentralised scheme, a sensor node becomes a *self-reporting* node upon the occurrence of externally triggered events. It then initiates information exchange to try to get extra slots from nodes in its neighborhood. In the centralised scheme, the central manager executes the function to distribute load among sensor nodes to prolong the lifetime of the network. In WinMS, a *self-reporting* node borrows extra slots from other nodes only for a period of time that is application specific. Nodes that lend their slots simply revert to their previous schedule once the loan time expires.

An advantage of WinMS is that its lightweight TDMA protocol provides energy-efficient management, data transport and local repair. Its systematic resource transfer function allows non-uniform and reactive sensing in different parts of a network, and it provides automatic self-configuration and self-stabilization both locally and globally by allowing the network to adapt to current network conditions without human intervention. A disadvantage of WinMS is that the initial setup cost for building a data gathering tree and node schedule is proportional to network density. However, this one-off cost can be tolerated because nodes maintain the gathered information throughout their lifetime in the network.

C. Management By Delegation

Management by delegation (MbD) allows management roles to be distributed across nodes in the network to ease the burden on a central manager. Delegation is achieved by intelligent agents or mobile agents. A mobile agent is a section of code that can distribute management tasks to be executed on nodes locally and returns the resulting data to the central manager [4]. Several protocols use delegation schemes to providing network management functions including Agilla [29], sectoral sweepers (SS) [30], mobile agent-based policy management [31], and intelligent agent-based power management [23].

1. Agilla

Agilla [29] is middleware that allows users to deploy mobile agents in a network to perform application-specific tasks. For example, in a fire tracking application,when a fire occurs in some region of a sensor network, agents will track the fire as it spreads, and will form a perimeter for the fire area. In Agilla, each sensor node can be monitored by multiple agents. Agilla provides a scheme that enables mobile agents to move themselves to desired locations as network conditions change. Each sensor node in the network maintains a tuple space that contains a set of predefined descriptors about that node. A node's tuple space can be shared by local agents and agents can register their interest in particular events by inserting a template tuple into the tuple space. When a node detects matching events, it updates its tuple space accordingly and reports the events to the agent. Thus, agents do not need to continually poll network states from sensor nodes.

2. Mobile Agent-Based Policy Management

A hierarchical mobile agent-based policy management for managing sensor networks is proposed in [31]. It allows the end user to pre-specify management policies, rules for the use of energy and computing power, that are enforced by mobile agents. Each rule consists of conditions and management operations to be carried out when the conditions are satisfied.

FIG. 3 shows the hierarchical architecture of the proposed system [31]. The system consists of three levels: Policy Manager (PM) at the highest level, Cluster Policy Agent (CPA), and Local Policy Agent (LPA). A PM manages multiple CPAs and adaptively reconfigures the network (locally or globally) when network conditions change. A CPA is the node with the best resources in a cluster and it manages multiple LPAs. An LPA manages a sensor node and also enforces local policies by analysing network dynamics (e.g., topology change), performing configuration, monitoring, filtering, and reporting. Policies are propagated from the PM to CPAs to LPAs, or from CPAs to LPAs. For example, when users want to execute a management function, the policy manager first transmits the policy for relevant management function to the access point. The access point transmits the policy to the cluster-head which transmits the policy to sensor nodes. The sensor node with data informs the agent and the agent returns the resulting sensor data to the cluster head. The agent returns to the access point if received data is not from the target node.

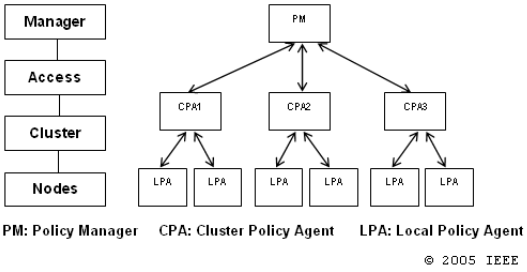

FIG. 3: Hierarchical architecture of mobile agent-based policy management. Reprinted with permission from [31].

The main advantage of this system is the adaptability and reconfigurability of network management since agents are active all the time. Furthermore, policy agents organised in a hierarchy can be used to perform network management functions either locally or globally. Users can flexibly inject new management functions into the system. The system ensures survivability by allowing an agent to replace another agent's management role in the event of failure. However, this method does not provide details for implementation of the proposed mobile agent-based management framework.

3. Sectoral Sweeper

Erdogan et al. [30] propose sectoral sweepers (SS) for managing a wireless sensor network. A sectoral sweeper is placed in the network to manage a region of the network. A sweeper acts as a distributed manager that directly disseminates tasks to sensor nodes within its region of responsibility by broadcasting. This approach reduces network energy consumption by eliminating the need for multi-hop transmissions to deliver a task to target nodes. Sensor nodes that are not involved in disseminating a task, may sleep to conserve their energy.

In the sectoral sweepers system (SS), the end user controls the network by determining the size of task region, resizing a task region, and terminating a task. A sweeper specifies the size of a task region Each task contains descriptions associated with a region and this allows sensor nodes to be assigned multiple tasks. For example, the end user can activate or deactivate nodes within a certain region in the network at any time. Sensor nodes use task descriptors and received signal strength to check if they are within a sweeper task region. Initially, sen-

sor nodes are in SLEEPING mode until they receive a ROUTE or ENGAGE task. They wake up regularly to check for tasks from a sleeper. When they receive a task, they switch to ENGAGE state if they have to perform the task specified by the sweeper, or switch to ROUTE state if they need to forward messages for other nodes to the sink. Nodes outside the task region can continue to sleep to conserve energy.

An advantage of the sectoral sweeper system is that it allows users to manage different parts of the network with independent or overlapping management functions. The distributed management approach reduces message implosion problems at the sink, reduces redundant data traffic by limiting the number of nodes involved in performing a task, and reduces latency between the sensor data polling and response arrival by eliminating the need for multiple hops to deliver tasks and task results from and to the sink. In addition, sleepers can be implemented above any MAC, network and transport layer protocol. The main drawback of SS is that it requires a manual configuration of sweeper placement. Since SS manages the network in a sectoral manner, users need to strategically place sweepers in the field.

4. Intelligent Agent-Based Power Management

Tynan et al. [23] have designed an intelligent agent-based power management system (IABP) using the Belief, Desire and Intention paradigm [40]. In IABP, beliefs represent states of sensor nodes that an agent holds to be true. Commitment rules (or desires) are pre-defined conditions to evaluate beliefs. If beliefs match commitment rules, the corresponding commitment management function (intention) will be executed. This agent-based approach is designed for applications where only a partial view of the state of the network as a whole can be known at any one location or time [23].

IABP agents make power management decisions locally based on requirements of an application [23]. By using agents, information exchange between nodes in a neighborhood in order to make a local decision can be eliminated since agents collect node data and process it to meet a specified goal. For example, the base station could inject a mobile agent into a sensor network to evaluate battery level of sensors in the network. This agent could also command nodes to reduce the sampling rate of sensors if their battery level is low. This scheme allows the base station to assess network states locally rather than gathering sensor node states to the base station.

Power management is also strongly related to other network attributes such as coverage, accuracy, battery longevity, and latency. The proposed agent-based approach [23] can perform complex decision making for various energy saving strategies. Users can specify desired sampling frequency, transmission range, and node mobility. Agents can be used to redirect traffic or change a link between nodes in the network to ensure a balance between energy conservation and network coverage. When a node's battery level is critical, the agent finds another nearby node that can forward data. End users can also control sampling frequency by commanding sensor nodes to transmit only when there is something worth reporting. The energy preserved by reducing transmissions allows a greater sampling rate of sensor nodes, which usually increases the accuracy of sensor data. However, when data polling rates are reduced, there is a risk of missing a crucial event. End users can command that nodes reduce their transmission power in order to conserve power. However, since reducing transmission power reduces communication range, this scheme may compromise network connectivity. The degree of agent mobility freedom allowed in the network can influence the latency of data collected from sensor nodes to the user.

D. Debugging Tools

1. Sympathy

Sympathy, is a debugging system developed to identify and localize the cause of failures in sensor network applications by collecting network performance metrics with minimal memory overhead, analysing the metrics to detect events, and identifying spatiotemporal context of the events [19]. Failures in sensor networks may be due to interactions among several nodes. Sympathy provides a debugging scheme to detect this type of failure by analysing the correlation between events. Sympathy collects metrics that represent states of the network and are gathered directly from the application. Sympathy defines metrics for inferring network health, fault detection (e.g. route flapping, next-hop selection, and packet loss), and other events specific to sensor networks. Sympathy updates metrics as they change and significant changes in metrics trigger events. For example, after the network is configured, nodes' initial neighborhood list, routes, and connectivity are known. Sympathy then tracks any occurrence of following events: missing node, isolated node, route change, neighbor list change, and link quality change.

Sympathy has two types of nodes: a Sympathy-sink and a Sympathy-node. The Sympathy-sink is a node that requests data from the Sympathy-node. The Sympathy-node is a sensor node in the network that collects and monitors network metrics, detects environmental events, and provides requested data to the sink. To minimize energy consumption, a Sympathy node may decide to transmit detected events selectively to the Sympathy sink. The Sympathy-sink uses a flooding approach to request sensor nodes to send their event data and current metric states. After receiving the metrics, the Sympathy-sink analyses them to process the event context. When the Sympathy-sink detects an event, it provides a temporal context by retrieving historical metrics associated with the node causing the trigger and also neighbors of the node. The Sympathy sink can probe the root cause of

the problem (event) by injecting a request to the network to collect the neighborhood information of that node. Once the Sympathy sink verifies the hypothesis of the root cause, it informs clients interested in that event. For example, if the next-hop-metric changes frequently, this event may indicate bad route configurations or network instability. In Sympathy, most event detection schemes require nodes only to send their metrics to their local one-hop neighbors. The sink, with global knowledge of the network, is only used to identify the missing-node event.

A strength of Sympathy is its ability to highlight failures and localize their cause by correlating detected events with metrics to determine their spatio-temporal context. Traditional debugging and fault detection techniques often assume that wireless nodes have unlimited resources and nodes fail as a result of local causes. Sympathy takes into account interactions among multiple nodes, provides a mechanism to analyse the context of an event, maintains network states, and identifies events of interest proactively. In these ways, Sympathy is superior to traditional debugging techniques. However, Sympathy does not provide automatic bug detection, it relies on historical data and post-mortem analysis of metrics in order to isolate the cause of failure. Furthermore, in detecting a fault, Sympathy may require nodes to exchange neighborhood lists. This approach is expensive in terms of energy consumption and delay associated with node communication. Moreover, Sympathy's flooding approach means that the Sympathy sink may have imprecise knowledge of global network states and so may make inaccurate analyses.

2. Two-Phase Self-Monitoring System

Hsin and Liu [18] propose an efficient distributed self-monitoring mechanism, called a two-phase self monitoring system (TP). TP is designed for monitoring and surveillance applications of sensor networks. In these applications, health monitoring of individual nodes is important for detecting malfunctioning nodes and intrusions that can result in the destruction of nodes. The TP system utilizes a two-phase timer scheme for local coordination and active probing. In the first phase, a node waits for updates from a neighbor. In the second phase, a node collaborates with its neighbors to clarify a condition in order to make a more accurate management decision.

In TP, fault detection is either explicit or implicit. Explicit fault detection is performed by nodes analysing sensor data and triggering an alarm if an event of interest occurs. For example, a node triggers an alarm if it records a temperature exceeding a pre-programmed threshold. This scheme offers low energy overhead as the base station expects nothing unless the nodes report event triggers. Implicit fault detection refers to the detection of node communication failures that may be due to energy depletion, intrusion, or environmental factors such as physical damage to nodes. To detect implicit faults, continuous monitoring of sensor nodes is required. TP uses a distributed scheme for monitoring node activity in which nodes perform both implicit (active monitoring) and explicit (passive monitoring) fault detection based on neighborhood information.

In TP, implicit fault detection is performed between neighbors and fault alarms are sent to the base station. Each node sends 'alive' update messages to its neighbors and also monitors its neighbors actively. A TP node uses a timer, and a node assumes its neighbor is dead if it has not heard from the neighbor for a predefined period of time. Since neighbors monitor each other, health monitoring information can be propagated throughout the network. However, this scheme fails when there is a network partition. Explicit fault detection is performed through neighborhood coordination and an alarm is sent to the base station only if nodes have high confidence that a fault has occurred. When a node detects an event, the node first exchanges information with its neighbors to probe the events accuracy before triggering an alarm.

Hsin and Liu [18] propose two performance metrics to evaluate TPs self-monitoring mechanism: false alarm probability and response delay. The former is the probability that a sensor has been deemed dead while it is actually alive. Response delay is the time between a fault incident and its detection. TP employs a timeout-based detection scheme for monitoring a node by maintaining two timers with values *T1* and *T2* respectively. For example, if node X receives no packets from node Y during the first timer *T1*, it activates the second timer *T2* before *T1* expires. Node X then consults with other neighbors regarding the status of node Y during *T2*. This local coordination allows node X to justify whether receiving no packets from node Y is because of noisy radio environment or node Y is actually dead.

The performance of TP is strongly influenced by the timer parameters *T1* and *T2*. A longer timeout can provide more accurate fault detection (smaller false alarm probability) but also leads to slower response (longer response delay). TP aims to optimize this trade-off. For a stable environment, TP uses analytical estimates as a guideline in determining optimal values for *T1* and *T2*. For a noisy environment, TP allows nodes to dynamically self-tune *T1* and *T2* values. Nodes use the average packet reception rate to evaluate environment conditions and then adjust *T1* and *T2* accordingly.

The main drawback of TP is that its responsiveness depends on the effectiveness of the routing protocol in propagating the alarm. In [18], a random access type MAC is used for simulating TP node behaviors. This protocol is prone to collisions and hence high energy consumption. Furthermore, TP has no synchronization scheme. Timers are only updated during packet receptions. Propagation delay in wireless links causes neighbors to get different packet reception times and this can result in timer drifts between neighbors. Moreover, TP is not scalable to a

very dense network, since each node's memory may be insufficient to record its neighborhood tables.

E. Visualisation Tool

1. MOTE-VIEW

MOTE-VIEW [22] is a visualisation tool designed for monitoring and managing sensor networks. The goals of MOTE-VIEW are to support users in analysing the large amounts of data generated by sensor networks, to monitor the health and status of individual sensor nodes and the network as a whole, and to present meaningful information to the end user. MOTE-VIEW uses a centralised-management approach in which all data and management processing is performed centrally by the server.

The MOTE-VIEW architecture consists of four layers. First, the Data Access Abstraction Layer is the database server interface that is used to retrieve sensor nodes data. Second, Node Abstraction Layer is an interface that stores sensor nodes' metadata (e.g. name, configuration, and calibration coefficient). This layer enables the end user to retrieve or change node parameter settings, such as radio frequency, power selection, sample rate, and custom calibration. Third, Conversion Abstraction Layer is responsible to calibrate and covert raw sensor data (digital readings) into engineering units that are understandable to the end user. Lastly, Visualization Abstraction Layer provides a textual and graphical display of sensory and link quality data in forms of spreadsheets, charts, or a network topology. Furthermore, this layer enables the end user to browse through historical data in a temporal context. For example, MOTE-VIEW creates animated movies of collected data.

In MOTE-VIEW, sensor nodes forward their data to the base station where it is collected by the central server that sits in the Data Access Abstraction Layer. The end-user makes management queries in the Visualization Abstraction Layer, such as querying the health or battery life expectation of a node, link quality, and network throughput. MOTE-VIEW employs a color coding technique to visualize the health of a node or a network. For example, if the base station receives no readings from a particular node for a period of time, the visualizer changes the color of the node from green to orange to indicate that the sensor readings are outdated.

MOTE-VIEW provides a plug-in facility to support modular extension for all of its four layers. It provides a data aggregation technique for presenting multiple and complex data sets in which it visualizes data from a specified set of nodes instead of individual node data. The modular design and advanced visualisation techniques of MOTE-VIEW allow it to scale as the network grows. Since the central server performs all post-mortem analysis of data collected by the sensor network, MOTE-VIEW removes the computational burden for management processing from resource-constrained sensor nodes. However, this passive monitoring approach does not allow networks to self-configure themselves in the event of node failures, which is desirable in most sensor network applications. Furthermore, the update interval of the network is often set to be slow to conserve node energy consumption, which compromises the accuracy of the reported network state.

F. Power Management Systems

Energy is one of the most important resources to be managed in a sensor network because sensor nodes are mostly battery powered and in many cases it is impractical to recharge these batteries [41]. Several researchers propose sensor management schemes that aim to provide control power consumption in WSNs in order to achieve energy-efficiency, including SenOS [9], AppSleep [24], and node-energy level management [25]. There are also other protocols that incorporate methods for reducing energy consumption. SPAN [42] and STEM [43] allow nodes to sleep if they are not involved in a forwarding task. LEACH [44] uses an in-network data aggregation method to filter redundant data in a dense sensor network and so reduce the number of transmissions.

1. SenOS

SenOS is a finite state machine based operating system that embeds a power management protocol into sensor nodes [9]. SenOS assumes that redundant nodes are placed in a cluster for alternating operation in order to prolong the clusters lifetime. It only keeps one node alive in a cluster for a period of time, while other nodes are put to sleep to conserve power. To achieve this, SenOS utilizes Dynamic power management (DPM) [37], a comprehensive algorithmic power management technique, to shut down nodes when not needed and to wake up nodes when necessary. DPM provides a policy for determining state transitions based on observed events in order to maximize energy efficiency. SenOS expresses state transitions produced by DPM in a finite state machine model and executes the power management of networked nodes based on this model. The state-driven SenOS framework is extensible to other sensor management protocols. However, the proposed network management protocol is limited to SenOS operating system-based sensor networks.

2. AppSleep

AppSleep is a stream-oriented power management protocol that extends the sleeping period of sensor nodes based on knowledge of scheduled communications in a network, while still providing low latency responses for

unscheduled communication when necessary [24]. AppSleep is designed for low duty cycle sensor network applications with bursty, delay insensitive data transmission. For example, a manufacturing monitoring application in which sensor nodes monitor the vibration of industrial equipment in order to predict future mechanical failures. In this application, sensor nodes periodically transmit data to the base station in a bursts. Users usually query the network after analysing the collected periodic data. AppSleep implements the power management function on the application layer to enable it to manage sleep schedules based on streams of packets instead of individual packets. Basically, AppSleep keeps nodes that are involved in multi-fragment data transfers awake, while putting other nodes to sleep.

In [24], Ramanathan et al. uses a hierarchical architecture with each cluster head acts as a manager that enforces AppSleep scheme. The cluster head periodically propagates a SYNC message that specifies when nodes in its cluster should wake up or sleep, the relative time for the cluster to return to sleep, and the waiting period for the next SYNC message. This scheme ensures that all nodes in the cluster sleep and wake at the same time and nodes stay awake during scheduled multi-hop transmissions. Guaranteeing that nodes on a path to the base station are awake eliminates the need for buffering because each active node forwards messages immediately to the next hop. AppSleep provides fault tolerance by allowing a node to stay awake and to request a SYNC message when it does not receive a scheduled SYNC packet or has not received a SYNC packet to join the network, for a new node. Increasing time synchronization frequency reduces clock drift but reduces the energy efficiency of AppSleep.

Ramanathan et al. [24] also propose Adaptive AppSleep allowing applications to adapt their behaviour according to latency and energy trade-offs specified by the user. To achieve this, AppSleep maintains a state machine and utilizes the periodic sending of SYNC messages to allow the user to specify the number of states (wake or sleep), the time spent in each state, and the initial sleep period of nodes in the network. For example, the user can command nodes in a cluster to be in an idle state after periodic data collection in order to be ready for any user query before sleeping again. Alternatively, the user can schedule a short sleep period immediately after a periodic data collection because during this period, unscheduled requests (event triggers or user queries) are unlikely to happen. The second scheme trades response latency for increased energy savings.

AppSleep's stream-based scheme is more energy efficient than packet-based protocol such as BMAC [45]. In BMAC, each node must listen to all control messages of its neighboring nodes to determine whether it is the intended receiver of the packet or not before going back to sleep. AppSleep is robust to changes in neighborhood density and it supports a scheme allowing applications to configure themselves to meet various latency requirements while maximizing energy efficiency. A limitation of AppSleep is that in order to take advantage of its energy saving scheme, applications must be able to tolerate long communication delay. Since AppSleep's operation depends on the routing layer, the chosen routing protocol should accommodate AppSleep characteristics: when active routes change, the routing protocol should be able to re-establish new routes at the beginning of wake period; and the routing protocol should ensure that selected routes for data transfers are active during the entire wake up period.

3. Node-Energy Level Management

Many different application tasks may be injected into a sensor network. It is desirable for the network to be able to accept or reject a task based on energy costs, merits, and objectives of the task in meeting overall user goals [25]. In a system-level approach, each application needs to have complete knowledge of all other applications and this results in a large traffic overhead for node communications. To address this issue, Boulis and Srivastava [25] propose a mechanism that solves the energy management problem at the node level instead of the system level, whereby nodes can make local decisions about accepting tasks. Applications do not need to directly communicate with each other.

The proposed node-energy level management evaluates three attributes in deciding whether to accept or reject a task: energy attributes, reward attributes, and admission policy attributes. Energy attributes are a list of service names, each containing parameters that define the service usage, that is the energy cost, of a specific task. These parameter values are pre-programmed when a task is created. Reward attributes specify the users priority and application reward: the level of information accuracy provided by a node in meeting overall management goals. For example, when a task is assigned to a node, the node evaluates the task's energy cost against its reward and arrives at a composite measure that is used to determine whether the task is beneficial to the user. Based on admission policies such as a node's remaining energy, nodes decides whether to accept or reject a task. Since nodes may receive multiple tasks or already be running a set of tasks, each node needs to keep a usage profile for each received task in order to measure accurately the energy saving gained by rejecting a certain task. A drawback of this scheme is that it increases the data storage requirements and computational cost of task execution.

G. Traffic Management Systems

1. Siphon

Siphon is an on-demand overload traffic management system [26]. It uses a small number of wireless multi-

radio virtual sinks to prevent congestion at or near base stations. Siphon proposes distributed algorithms that provide virtual sink discovery and selection, congestion detection, traffic redirection, and congestion avoidance in the secondary radio network. Virtual sinks (VSs) are intermediaries between sensor nodes and base stations in the network and are responsible for redirecting data from regions of the sensor network that have increasing traffic loads.

Siphon uses an in-band signaling approach to allow nodes with data overload to discover local VSs that could be multiple hops away. Siphon does this by piggybacking a signature byte into a control packet that is periodically advertised by the base station. The signature byte contains the scope (hop-count) of a VS and this information is used to control the visibility of the VS to nodes in its neighborhood. Siphon uses two techniques to detect congestion: node-initiated congestion detection and physical sink initiated post-facto congestion control. Siphon uses a CSMA-based congestion detection technique [46] to detect local congestion, whereby a VS redirects the traffic of nodes within its visibility when their data generation rates exceed a predefined threshold. Siphon also performs post-mortem analysis of event data at the base station. Siphon activates VS signaling originating at the base station if the reliability of the network degrades below a predefined threshold. This scheme is useful for preventing congestion close to the base station. Siphon uses a secondary radio network to propagate VS signaling in a timely manner without interfering with the operation of the primary-radio network. In addition, a VS constantly monitors its congestion level on both primary and secondary radio channels to avoid congestion at any channel. When a channel is overloaded, a VS may not advertise its existence or may reduce its service scope. If both channels are overloaded, Siphon uses a traditional fallback mechanism such as controlling data rates at the source node and at forwarding nodes to ease congestion.

Advantages of Siphon are that it prevents the funneling effect at a base station from the many-to-one traffic pattern and it is able to accommodate various traffic load demands in a network. Furthermore, the VS second-radio infrastructure allows traffic redirection in a timely manner without degrading the performance of forwarding operations in the primary-radio network. Lastly, VS can detect congestion locally and globally. Global congestion detection performed by the base station is beneficial for avoiding premature funneling problems. Disadvantages of Siphon are that it requires pre-deployment expertise in determining the optimal VS scope (value of hop-count) under different conditions, distributing or selectively placing virtual sinks across the sensor field, and determining the minimum number of virtual sinks required to achieve optimum traffic management performance. In addition, the monetary cost for deploying a multi-radio sensor platform is much higher than a single-radio platform. A VS with a large number of hop counts may create a funneling problem at the VS. In contrast, a VS with smaller hop counts provides shorter redirection paths from congested nodes to it and hence improves delivery latency and energy consumption. However, it requires the application to deploy more VSs to cover the network and this increases the cost of deploying the network, defeating the benefit of using fewer VSs. Furthermore, some nodes may not be covered by any VS as a result of random distribution or limited availability of VSs in a sensor network.

2. DSN Resource Management

Zhang et al. [27] propose a resource management technique for task oriented distributed sensor networks (DSN) that avoids network congestion while meeting the overall objectives of the network. They propose a 'per-flow' method to analyse data streams among nodes at different hierarchical levels of the DSN. In DSNs, the decision stations act as managers for each hierarchical level. Each manager gathers data from nodes in its level, processes their data, and sends the resulting information to the next level manager for further processing. Thus, the information produced at lower levels affects the management decision making process at higher levels, and low level information can be used to meet global objectives set by the highest level.

The per-flow method provides a set of measures for evaluating the importance (degradation and relevance) of incoming data associated with each link at a node. It uses fuzzy logic techniques to determine suitable degradation and relevance values. Basically, each node in the network assigns three weight measures to each of its network links: 1) data quality measured at the node's parent, 2) data timeliness: latency incurred for data to arrive at the node, and 3) data significance in achieving the overall DSN objectives, measured at the node's child. These measurements are used to allocate resources among nodes in a network in which higher weight links get priority over lower weight links. This scheme can reduce data flows in the network and hence prevent network congestion.

An advantage of per-flow priority-based resource management is that it integrates and propagates relevant information from the network for achieving the overall objectives of a DSN, hence providing intelligent resource management and congestion avoidance. A limitation of the proposed technique is that its effectiveness depends on finding reliable data weight values, which are prone to WSN uncertainties.

H. Applications with Network Management Functions

In addition to applications dedicated to network management, many WSN applications incorporate network management functions. For example, TinyDB [21] is a query-based processing system for extracting informa-

tion from sensors in a network. It maintains meta-data describing the types of sensor readings available in the network. It also provides a network topology map to manage the network by monitoring connectivity among nodes, maintaining routing tables, and ensuring that every sensor node in the network delivers its data to the end user efficiently [21]. A limitation of TinyDB from a management perspective, is that it requires the human manager to control network management operations manually and to interpret the collected management information. Hence, it is necessarily a reactive system in which nodes cannot manage their own behavior based on their sensor readings and topology changes are difficult to capture [1]. Zhao et al. [47] utilize in-network aggregation of network states to provide an abstraction of sensor network health, such as a residual energy scan that represents the remaining energy level of sensor nodes [48].

In WSNs, sensor nodes may have overlapping coverage areas and so these nodes produce redundant data. It is beneficial for energy conservation to shut down redundant nodes temporarily or reduce their reporting rates. Tilak et al. [49] propose a congestion avoidance scheme that modifies sensor node reporting rates when the data collected has met a desired reliability. This scheme can reduce the number of transmissions in the sensor network and so prevent network congestion.

Perillo and Heinzelman [28] propose a sensor management optimization method to maximize the lifetime of a sensor network while meeting a desired level of reliability. The balance is achieved by selecting and scheduling a set of nodes to be active in monitoring the environment and determining how long and how the data from these nodes should be routed in the network. Redundant nodes are turned off during this time. The management scheme is designed for event-driven applications in which event triggers occur infrequently and only for a short period of time. A set of active nodes is valid if the total bandwidth requirement of the set is within the network's bandwidth capacity and the set meets the desired reliability level. Furthermore, the management scheme models the scheduling and routing requirement as a generalized maximum flow graph problem in order to ensure that active nodes are routed as often as possible. A drawback of the proposed scheme is that the optimization algorithm for solving the graph problem is computationally too complex and expensive for resource-constrained sensor nodes.

V. SUMMARY

The sensor network management systems reviewed in this chapter have been designed from many different management perspectives: network-health monitoring, fault-detection, traffic management, congestion avoidance, power management, and resource management. The systems are characterized by their power consumption, memory consumption, bandwidth consumption, fault tolerance, adaptability, and scalability. None of the reviewed systems provides a fully integrated view of all sensor network management design factors. Furthermore, most of these systems incorporate management functions within application protocols. The development of general purpose network management layer protocols is a challenging problem and remains a largely unexplored area for wireless sensor networks. Another significant open problem is the development of management policies and expressive languages or metadata for representing management policies and for representing the information exchanged between sensor nodes, managers, and end users.

Acknowledgments

This research is partially funded by an Australian Postgraduate Award, and a scholarship from the School of Computer Science & Software Engineering at the University of Western Australia.

[1] M. Welsh and G. Mainland, "Programming Sensor Networks Using Abstract Regions," in *Proc. USENIX NSDI Conf.*, Mar. 2004.

[2] A. Boulis, C. Han, and M. B. Srivastava, "Design and Implementation of a Framework for Efficient and Programmable Sensor Networks," in *Proc. ACM MobiSys Conf.*, May 2003.

[3] B. Deb, S. Bhatnagar, and B. Nath, "A Topology Discovery Algorithm for Sensor Networks with Applications to Network Management," Tech. Rep. DCS-TR-441, Rutgers University, May 2001.

[4] L.B. Ruiz, J.M. Nogueira, and A.A.F. Loureiro, "MANNA: A Management Architecture for Wireless Sensor Networks," *IEEE Communications Magazine*, vol. 41, no. 2, pp. 116–125, 2003.

[5] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson, "Wireless Sensor Networks for Habitat Monitoring," in *Proc. ACM WSNA Conf.*, Sep. 2002.

[6] D. Estrin, L. Girod, G. Pottie, and M. Srivastava, "Instrumenting the World with Wireless Sensor Networks," in *Proc. IEEE ICASSP Conf.*, May 2001.

[7] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless Sensor Networks: A Survey," *Computer Networks*, vol. 38, no. 4, pp. 393–422, 2002.

[8] N. Xu, S. Rangwala, K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan, and D. Estrin, "A Wireless Sensor Network for Structural Monitoring," in *Proc. ACM SenSys Conf.*, Nov. 2004.

[9] T.H. Kim and S. Hong, "Sensor Network Management Protocol for State-Driven Execution Environment," in *Proc. ICUC Conf.*, Oct. 2003.

[10] J. Heidemann, F. Silva, C. Intanagonwiwat, R. Govin-

dan, D. Estrin, and D. Ganesan, "Building Efficient Wireless Sensor Networks with Low-Level Naming," in *Proc. ACM SOSP Conf.*, Oct. 2001.

[11] C. Intanagonwiwat, D. Estrin, R. Govindan, and J. Heidemann, "Impact of Network Density on Data Aggregation in Wireless Sensor Networks," Tech. Rep. 01-750, University of Southern California, Nov. 2001.

[12] H. Song, D. Kim, K. Lee, and J. Sung, "Upnp-Based Sensor Network Management Architecture," in *Proc. ICMU Conf.*, Apr. 2005.

[13] B. Deb, S. Bhatnagar, and B. Nath, "Wireless Sensor Networks Management," http://www.research.rutgers.edu/∼bdeb/sensor_networks.html, 2005.

[14] B. Deb, S. Bhatnagar, and B. Nath, "STREAM: Sensor Topology Retrieval at Multiple Resolutions," *Kluwer Journal of Telecommunications Systems*, vol. 26, no. 2, pp. 285–320, 2004.

[15] W. Liu, Y. Zhang, W. Lou, and Y. Fang, "Managing Wireless Sensor Network with Supply Chain Strategy," in *Proc. IEEE QSHINE Conf.*, Oct. 2004.

[16] G. Tolle and D. Culler, "Design of an Application-Cooperative Management System for Wireless Sensor Networks," in *Proc. EWSN*, Feb. 2005.

[17] W. Louis Lee, A. Datta, and R. Cardell-Oliver, "WinMS: **Wi**reless Sensor **n**etwork-Management **s**ystem, An Adaptive Policy-based Management for Wireless Sensor Networks," Tech. Rep. UWA-CSSE-06-001, The University of Western Australia, June 2006.

[18] C. Hsin and M. Liu, "A Two-Phase Self-Monitoring Mechanism for Wireless Sensor Networks," *Journal of Computer Communications special issue on Sensor Networks*, vol. 29, no. 4, pp. 462–476, 2006.

[19] N. Ramanathan, E. Kohler, and D. Estrin, "Towards a Debugging System for Sensor Networks," *International Journal for Network Management*, vol. 15, no. 4, pp. 223–234, 2005.

[20] L.B. Ruiz, I.G. Siqueira, L.B. e Oliveira, H.C. Wong, J.M.S. Nogueira, and A.A.F. Loureiro, "Fault Management in Event-Driven Wireless Sensor Networks," in *Proc. ACM MSWiM Conf.*, Oct. 2004.

[21] S. Madden, J. Hellerstein, and W. Hong, "TinyDB: In-Network Query Processing in TinyOS," http://telegraph.cs.berkeley.edu/tinydb/tinydb.pdf, 2005.

[22] M. Turon, "MOTE-VIEW: A Sensor Network Monitoring and Management Tool," in *Proc. IEEE EmNetS-II Workshop*, May 2005.

[23] R. Tynan, D. Marsh, D. OKane, and G. M. P. OHare, "Agents for Wireless Sensor Network Power Management," in *Proc. IEEE ICPPW Conf.*, June 2005.

[24] N. Ramanathan and M. Yarvis, "A Stream-oriented Power Management Protocol for Low Duty Cycle Sensor Network Applications," in *Proc. IEEE EmNetS-II Workshop*, May 2005.

[25] A. Boulis and M.B. Srivastava, "Node-level Energy Management for Sensor Networks in the Presence of Multiple Applications," in *Proc. IEEE PerCom Conf.*, Mar. 2003.

[26] C-Y. Wan, S. B. Eisenman, A. T. Campbell, and J. Crowcrof, "Siphon: Overload Traffic Management using Multi-radio Virtual Sinks in Sensor Networks," in *Proc. ACM SenSys Conf.*, Nov. 2005.

[27] J. Zhang, E.C. Kulasekere, K. Premaratne, and P.H. Bauer, "Resource Management of Task Oriented Distributed Sensor Networks," in *Proc. IEEE ICASSP Conf.*, May 2001.

[28] M. Perillo and W.B. Heinzelman, "Providing Application QoS through Intelligent Sensor Management," in *Proc. IEEE SNPA Conf.*, May 2003.

[29] C. Fok, G. Roman, and C. Lu, "Mobile Agent Middleware for Sensor Networks: An Application Case Study," in *Proc. IEEE ICDCS Conf.*, June 2005.

[30] A. Erdogan, E. Cayirci, and V. Coskun, "Sectoral Sweepers for Sensor Node Management and Location Estimation in Adhoc Sensor Networks," in *Proc. IEEE MILCOM Conf.*, Oct. 2003.

[31] Z. Ying and X. Debao, "Mobile Agent-based Policy Management for Wireless Sensor Networks," in *Proc. IEEE WCNM Conf.*, Sep. 2005.

[32] L. B. Ruiz, *MANNA: A Management Architecture for Wireless Sensor Networks*, Ph.d. dissertation, Federal Univ. of Minas Gerais, Belo Horizonte, MG, Brazil, Dec. 2003.

[33] L. Subramanian and R.H. Katz, "An Architecture for Building Self Configurable Systems," in *Proc. IEEE MobiHOC Conf.*, Aug. 2000.

[34] J. Staddon, D. Balfanz, and G. Durfee, "Efficient Tracing of Failed Nodes in Sensor Networks," in *Proc. ACM WSNA Conf.*, Sep. 2002.

[35] L.B. Ruiz, F.B. Silva, T.R.M. Braga, J.M.S. Nogueira, and A.A.F. Loureiro, "On Impact of Management in Wireless Sensor Networks," in *Proc. IEEE/IFIP NOMS*, Apr. 2004.

[36] L.B. Ruiz, T.R.M. Braga, F.A. Silva, H.P. Assuncao, J.M.S. Nogueira, and A.A.F. Loureiro, "On the Design of a Self-Managed Wireless Sensor Network," *IEEE Communications Magazine*, vol. 43, no. 8, pp. 95–102, 2005.

[37] A. Sinha and A. Chandrakasan, "Dynamic Power Management in Wireless Sensor Networks," *IEEE Design and Test of Computers*, vol. 18, no. 2, pp. 62–74, 2001.

[38] J. Kulik, W. Heinzelman, and H. Balakrishnan, "Negotiation-Based Protocols for Disseminating Information in Wireless Sensor Networks," *ACM Wireless Networks*, vol. 8, no. 2-3, pp. 169–185, 2002.

[39] W. Louis Lee, A. Datta, and R. Cardell-Oliver, "Flexi-MAC: A Flexible TDMA-based MAC Protocol for Fault-tolerant and Energy-efficient Wireless Sensor Networks," in *To Appear in Proc. IEEE ICON Conf.*, Sep. 2006.

[40] H. Zhang and J. C. Hou, "Maintaining Sensing Coverage and Connectivity in Large Sensor Networks," in *Proc. NSF TAWN Conf.*, June 2004.

[41] U. Cetintemel, A. Flinders, and Y. Sun, "Power-Efficient Data Dissemination in Wireless Sensor Networks," in *Proc. ACM MobiDE Conf.*, Sep. 2003.

[42] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris, "Span: An Energy-Efficient Coordination Algorithm for Topology Maintenance in Ad Hoc Wireless Networks," in *Proc. ACM MobiCom Conf.*, Aug. 2000.

[43] C. Schurgers, V. Tsiatsis, S. Ganeriwal, and M. Srivastava, "Optimizing sensor networks in the energy-latency-density design space," *IEEE Transactions on Mobile Computing*, vol. 1, no. 1, Jan. 2002.

[44] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "An application-specific protocol architecture for wireless microsensor networks," *IEEE Transactions on Wireless Communications*, vol. 1, no. 4, Oct. 2002.

[45] J. Polastre, J. Hill, and D. Culler, "Versatile Low Power Media Access for Wireless Sensor Networks," in *Proc.*

ACM SenSys Conf., Nov. 2004.

[46] C-Y. Wan, S. B. Eisenman, and A. T. Campbell, "CODA: COngestion Detection and Avoidance in Sensor Networks," in *Proc. ACM SenSys Conf.*, Nov. 2003.

[47] Y.J. Zhao, R. Govindan, and D. Estrin, "Residual Energy Scan for Monitoring Sensor Networks," in *Proc. IEEE WCNC Conf.*, Mar. 2002.

[48] C. Intanagonwiwat, D. Estrin, and R. Gonvindan, "Impact of Network Density on Data Aggregation in Wireless Sensor Networks," in *Proc. IEEE ICDCS Conf.*, Apr. 2001.

[49] S. Tilak, N. B. Abu-Ghazaleh, and W. Heinzelman, "Infrastructure tradeoffs for sensor networks," in *Proc. ACM WSNA Conf.*, Sep. 2002.