Optimization Approaches in Wireless Sensor Networks

Arslan Munir and Ann Gordon-Ross
Department of Electrical and Computer Engineering
University of Florida, Gainesville, Florida, USA

1. Introduction

Advancements in silicon technology, micro-electro-mechanical systems (MEMS), wireless communications, and digital electronics have led to the proliferation of wireless sensor networks (WSNs) in a wide variety of application domains including military, health, ecology, environment, industrial automation, civil engineering, and medical. This wide application diversity combined with complex sensor node architectures, functionality requirements, and highly constrained and harsh operating environments makes WSN design very challenging.
One critical WSN design challenge involves meeting *application requirements* such as lifetime, reliability, throughput, delay (responsiveness), etc. for myriad of application domains. Furthermore, WSN applications tend to have competing requirements, which exacerbates design challenges. For example, a high priority security/defense system may have both high responsiveness and long lifetime requirements. The mechanisms needed for high responsiveness typically drain battery life quickly, thus making long lifetime difficult to achieve given limited energy reserves.
Commercial off-the-shelf (COTS) sensor nodes have difficulty meeting application requirements due to the generic design traits necessary for wide application applicability. COTS sensor nodes are mass-produced to optimize cost and are not specialized for any particular application. Fortunately, COTS sensor nodes contain *tunable parameters* (e.g., processor voltage and frequency, sensing frequency, etc.) whose values can be specialized to meet application requirements. However, optimizing these tunable parameters is left to the application designer.
Optimization techniques at different design levels (e.g., sensor node hardware and software, data link layer, routing, operating system (OS), etc.) assist designers in meeting application requirements. WSN optimization techniques can be generally categorized as *static* or *dynamic*. Static optimizations optimize a WSN at deployment time and remain fixed for the WSN's lifetime. Whereas static optimizations are suitable for stable/predictable applications, static optimizations are inflexible and do not adapt to changing application requirements and environmental stimuli. Dynamic optimizations provide more flexibility by continuously optimizing a WSN/sensor node during runtime, providing better adaptation to changing application requirements and actual environmental stimuli.
This chapter introduces WSNs from an optimization perspective and explores optimization strategies employed in WSNs at different design levels to meet application requirements

Design-level	Optimizations
Architecture-level	bridging, sensorweb, tunneling
Component-level	parameter-tuning (e.g., processor voltage and frequency, sensing frequency), MDP-based dynamic optimization
Data Link-level	load balancing and throughput, power/energy
Network-level	query dissemination, data aggregation, real-time, network topology, resource adaptive, dynamic network reprogramming
Operating System-level	event-driven, dynamic power management, fault-tolerance

Table 1. Optimizations (discussed in this chapter) at different design-levels.

as summarized in Table 1. We present a typical WSN architecture and architectural-level optimizations in Section 2. We describe sensor node component-level optimizations and tunable parameters in Section 3. Next, we discuss data link-level Medium Access Control (MAC) optimizations and network-level routing optimizations in Section 4 and Section 5, respectively, and operating system-level optimizations in Section 6. After presenting these optimization techniques, we focus on dynamic optimizations for WSNs. There exists much previous work on dynamic optimizations e.g., (Brooks & Martonosi, 2000); (Hamed et al., 2006); (Hazelwood & Smith, 2006); (Hu et al., 2006), but most previous work targets the processor or cache subsystem in computing systems. WSN dynamic optimizations present additional challenges due to a unique design space, stringent design constraints, and varying operating environments. We discuss the current state-of-the-art in dynamic optimization techniques in Section 7 and propose a Markov Decision Process (MDP)-based dynamic optimization methodology for WSNs to meet application requirements in the presence of changing environmental stimuli in Section 8. Numerical results validate the optimality of our MDP-based methodology and reveal that our methodology more closely meets application requirements as compared to other feasible policies.

2. Architecture-level Optimizations

Fig. 1 shows an integrated WSN architecture (i.e., a WSN integrated with external networks) capturing architecture-level optimizations. Sensor nodes are distributed in a *sensor field* to observe a phenomenon of interest (i.e., environment, vehicle, object, etc.). Sensor nodes in the sensor field form an ad hoc wireless network and transmit the sensed information (data or statistics) gathered via attached sensors about the observed phenomenon to a base station or *sink node*. The sink node relays the collected data to the remote requester (user) via an arbitrary computer communication network such as a gateway and associated communication network. Since different applications require different communication network infrastructures to efficiently transfer sensed data, WSN designers can optimize the communication architecture by determining the appropriate topology (number and distribution of sensors within the WSN) and communication infrastructure (e.g., gateway nodes) to meet the application's requirements.
An infrastructure-level optimization called *bridging* facilitates the transfer of sensed data to remote requesters residing at different locations by connecting the WSN to external networks such as Internet, cellular, and satellite networks. Bridging can be accomplished by overlaying a sensor network with portions of the IP network where gateway nodes encapsulate sensor

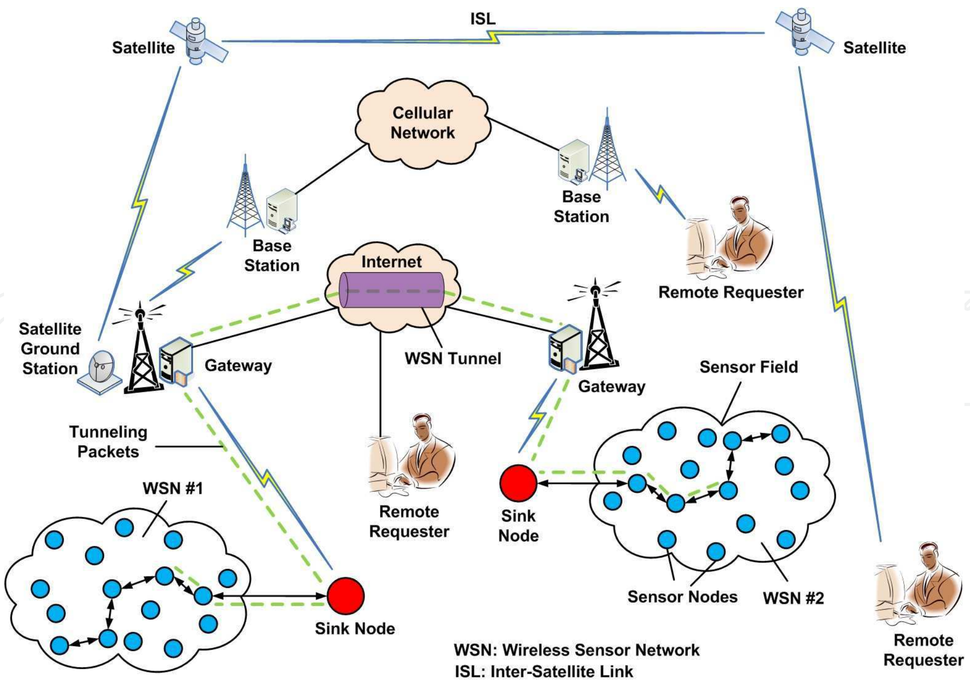


Fig. 1. Wireless sensor network architecture.

node packets with transmission control protocol or user datagram protocol/internet protocol (TCP/IP or UDP/IP).

Since sensor nodes can be integrated with the Internet via bridging, this WSN-Internet integration can be exploited to form a *sensor web*. In a sensor web, sensor nodes form a web view where data repositories, sensors, and image devices are discoverable, accessible, and controllable via the World Wide Web (WWW). The sensor web can use service-oriented architectures (SoAs) or sensor web enablement (SWE) standards (Mahalik, 2007). SoAs leverage extensible markup language (XML) and simple object access protocol (SOAP) standards to describe, discover, and invoke services from heterogeneous platforms. SWE is defined by the OpenGIS Consortium (OGC) and consists of specifications describing sensor data collection and web notification services. An example application for a sensor web may consist of a client using WSN information via sensor web queries. The client receives responses either from real-time sensors registered in the sensor web or from existing data in the sensor data base repository. In this application, clients can use WSN services without knowledge of the actual sensor nodes' locations.

Another WSN architectural optimization is *tunneling*. Tunneling connects two WSNs by passing internetwork communication through a gateway node that acts as a WSN extension and connects to an intermediate IP network. Tunneling enables construction of large virtual WSNs using smaller WSNs (Karl & Willig, 2005).

3. Sensor Node Component-level Optimizations

COTS sensor nodes provide optimization opportunities at the component-level via tunable parameters (e.g., processor voltage and frequency, sensing frequency, duty cycle, etc.), whose values can be specialized to meet varying application requirements. Fig. 2 depicts a sensor node's main components such as a power unit, storage unit, sensing unit, processing unit,

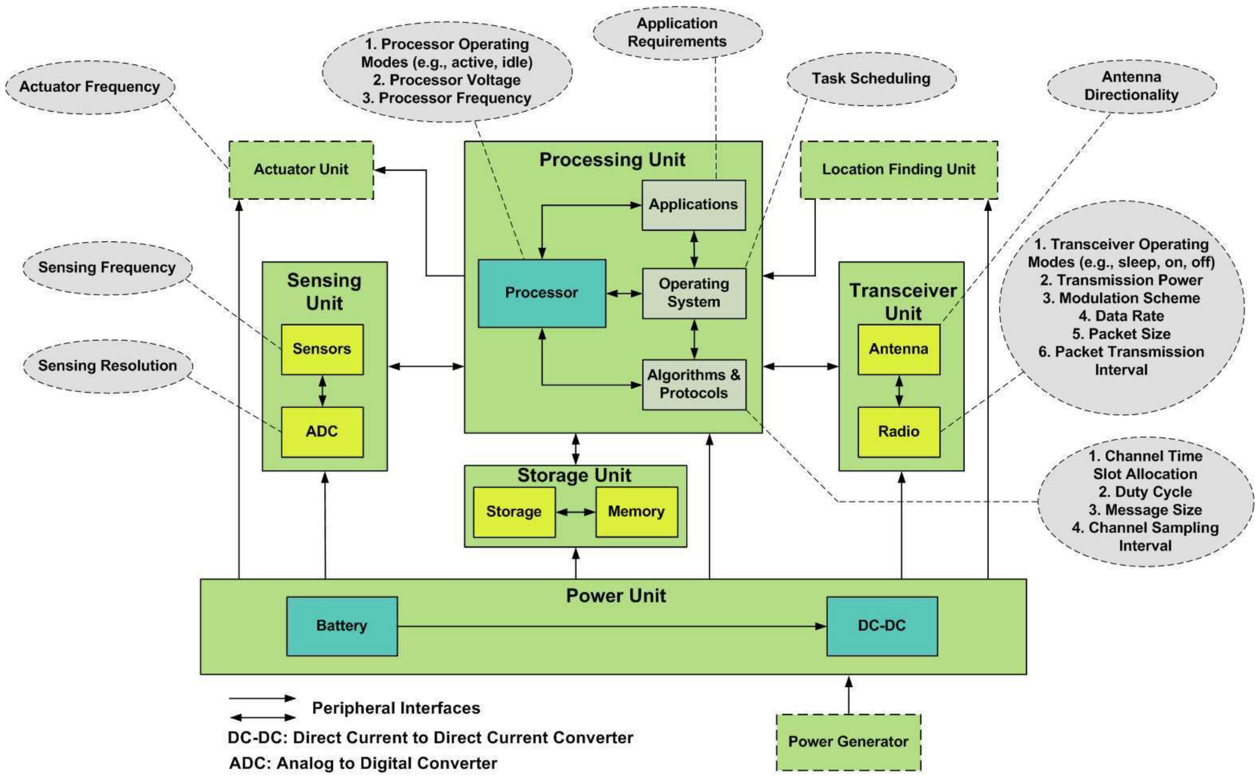


Fig. 2. Sensor node architecture with tunable parameters.

and transceiver unit along with potential tunable parameters associated with each component (Karl & Willig, 2005). In this section, we discuss these components and associated tunable parameters.

3.1 Sensing Unit

The sensing unit senses the phenomenon of interest using sensors and an analog to digital converter (ADC). The sensing unit's tunable parameters can control power consumption by changing the sensing frequency and the speed-resolution product of the ADC. Sensing frequency can be tuned to provide constant sensing, periodic sensing, and/or sporadic sensing. In constant sensing, sensors sense continuously and sensing frequency is limited only by the sensor hardware's design capabilities. Periodic sensing consumes less power than constant sensing because periodic sensing is duty-cycle based where the sensor node takes readings after every T seconds. Sporadic sensing consumes less power than periodic sensing because sporadic sensing is typically event-triggered by either external (e.g., environment) or internal (e.g., OS- or hardware-based) interrupts. The speed-resolution product of the ADC can be tuned to provide high speed-resolution with higher power consumption (e.g., seismic sensors use 24-bit converters with a conversion rate on the order of thousands of samples per second) or low speed-resolution with lower power consumption.

3.2 Processing Unit

The processing unit consists of a processor (e.g., Intel's Strong ARM (StrongARM, 2010), Atmel's AVR (ATMEL, 2009)) whose main tasks include controlling sensors, gathering and processing sensed data, executing WSN applications, and managing communication protocols and algorithms in conjunction with the operating system. The processor's tunable parameters include processor voltage and frequency, which can be specialized to meet power budget and throughput requirements. The processor can also switch between different operating modes (e.g., active, idle, sleep) to conserve energy. For example, the Intel's StrongARM consumes 75 mW in idle mode, 0.16 mW in sleep mode, and 240 mW and 400 mW in active mode while operating at 133 MHz and 206 MHz, respectively.

3.3 Transceiver Unit

The transceiver unit consists of a radio (transceiver) and an antenna, and is responsible for communicating with neighboring sensor nodes. The transceiver unit's tunable parameters include modulation scheme, data rate, transmit power, and duty cycle. The radio contains different operating modes (e.g., transmit, receive, idle, and sleep) for power management purposes. The sleep state provides the lowest power consumption, but switching from the sleep state to the transmit state consumes a large amount of power. The power saving modes (e.g., idle, sleep) are characterized by their power consumption and latency overhead (time to switch to transmit or receive modes). Power consumption in the transceiver unit also depends on the distance to the neighboring sensor nodes and transmission interferences (e.g., solar flare, radiation, channel noise).

3.4 Storage Unit

Sensor nodes contain a storage unit for temporary data storage since immediate data transmission is not always possible due to hardware failures, environmental conditions, physical layer jamming, and energy reserves. A sensor node's storage unit typically consists of Flash and static random access memory (SRAM). Flash is used for persistent storage of application code and text segments whereas SRAM is for run-time data storage. One potential optimization uses an extremely low-frequency (ELF) Flash file system, which is specifically adapted for sensor node data logging and operating environmental conditions. Storage unit optimization challenges include power conservation and memory resources (limited data and program memory, e.g., the Mica2 sensor node contains only 4 KB of data memory (SRAM) and 128 KB of program memory (Flash)).

3.5 Actuator Unit

The actuator unit consists of actuators (e.g., mobilizer, camera pan tilt), which enhance the sensing task. Actuators open/close a switch/relay to control functions such as camera or antenna orientation and repositioning sensors. Actuators, in contrast to sensors which only sense a phenomenon, typically affect the operating environment by opening a valve, emitting sound, or physically moving the sensor node. The actuator unit's tunable parameter is actuator frequency, which can be adjusted according to application requirements.

3.6 Location Finding Unit

The location finding unit determines a sensor node's location. Depending on the application requirements and available resources, the location finding unit can either be global positioning system (GPS)-based or ad hoc positioning system (APS)-based. The GPS-based location finding unit is highly accurate, but has high monetary cost and requires direct line of sight between the sensor node and satellites. The APS-based location finding unit determines a sensor node's position with respect to *landmarks*. Landmarks are typically GPS-based position-aware sensor nodes and landmark information is propagated in a multi-hop fashion. A sensor node in direct communication with a landmark estimates its distance from a landmark based on the received signal strength. A sensor node two hops away from a landmark estimates its distance based on the distance estimate of a sensor node one hop away from a landmark via message propagation. When a sensor node has distance estimates to three or more landmarks, the sensor node computes its own position as a centroid of the landmarks.

3.7 Power Unit

The power unit supplies power to a sensor node and determines a sensor node's lifetime. The power unit consists of a battery and a DC-DC converter. The electrode material and the diffusion rate of the electrolyte's active material affect the battery capacity. The DC-DC converter provides a constant supply voltage to the sensor node.

4. Data Link-level Medium Access Control Optimizations

Data link-level medium access control (MAC) manages the shared wireless channel and establishes data communication links between sensor nodes. Traditional MAC schemes emphasize high quality of service (QoS) (Rappaport, 1996) or bandwidth efficiency (Abramson, 1985); (IEEE Standards, 1999), however, WSN platforms have different priorities (Sohraby et al., 2007) thus inhibiting the straight forward adoption of existing MAC protocols (Chandrakasan et al., 1999). For example, since WSN lifetime is typically an important application requirement and batteries are not easily interchangeable/rechargeable, energy consumption is a primary design constraint for WSNs. Similarly, since the network infrastructure is subject to changes due to dying nodes, self-organization and failure recovery is important. To meet application requirements, WSN designers tune MAC layer protocol parameters (e.g., channel access schedule, message size, duty cycle, and receiver power-off, etc.). This section discusses MAC protocols for WSNs with reference to their tunable parameters and optimization objectives.

4.1 Load Balancing and Throughput Optimizations

MAC layer protocols can adjust wireless channel slot allocation to optimize throughput while maintaining the traffic load balance between sensor nodes. A *fairness* index measures load balancing or the uniformity of packets delivered to the sink node from all the senders. For the perfectly uniform case (ideal load balance), the fairness index is 1. MAC layer protocols that adjust channel slot allocation for load balancing and throughput optimizations include Traffic Adaptive Medium Access Protocol (TRAMA) (Rajendran et al., 2003), Berkeley Media Access Control (B-MAC) (Polastre et al., 2004), and Zebra MAC (Z-MAC) (Rhee et al., 2005).
TRAMA is a MAC protocol that adjusts channel time slot allocation to achieve load balancing while focusing on providing collision free medium access. TRAMA divides the channel access into random and scheduled access periods and aims to increase the utilization of the scheduled access period using time division multiple access (TDMA). TRAMA calculates a Message-Digest algorithm 5 (MD5) hash for every one-hop and two-hop neighboring sensor nodes to determine a node's priority. Experiments comparing TRAMA with both contention-based protocols (IEEE 802.11 and Sensor-MAC (S-MAC) (Ye et al., 2002)) as well as a scheduled-based protocol (Node-Activation Multiple Access (NAMA) (Bao & Garcia-Luna-Aceves, 2001)) revealed that TRAMA achieved higher throughput than contention-based protocols and comparable throughput with NAMA (Raghavendra et al., 2004).
B-MAC is a carrier sense MAC protocol for WSNs. B-MAC adjusts the duty cycle and time slot allocation for throughput optimization and high channel utilization. B-MAC supports

on-the-fly reconfiguration of the MAC backoff strategy for performance (e.g., throughput, latency, power conservation) optimization. Results from B-MAC and S-MAC implementation on TinyOS using Mica2 motes indicated that B-MAC outperformed S-MAC by 3.5x on average (Polastre et al., 2004). No sensor node was allocated more than 15% additional bandwidth as compared with other nodes, thus ensuring fairness (load balancing).

Z-MAC is a hybrid MAC protocol that combines the strengths of TDMA and carrier sense multiple access (CSMA) and offsets their weaknesses. Z-MAC allocates time slots at sensor node deployment time by using an efficient channel scheduling algorithm to optimize throughput, but this mechanism requires high initial overhead. A time slot's *owner* is the sensor node allocated to that time slot and all other nodes are called *non-owners* of that time slot. Multiple owners are possible for a given time slot because Z-MAC allows any two sensor nodes beyond their two-hop neighborhoods to own the same time slot. Unlike TDMA, a sensor node may transmit during any time slot but slot owners have a higher priority. Experimental results from Z-MAC implementation on both ns-2 and TinyOS/Mica2 indicated that Z-MAC performed better than B-MAC under medium to high contention but exhibited worse performance than B-MAC under low contention (inherits from TDMA-based channel access). The fairness index of Z-MAC was between 0.7 and 1, whereas that of B-MAC was between 0.2 to 0.3 for a large number of senders (Rhee et al., 2005).

4.2 Power/Energy Optimizations

MAC layer protocols can adapt their transceiver operating modes (e.g., sleep, on and off) and duty cycle for reduced power and/or energy consumption. MAC layer protocols that adjust duty cycle for power/energy optimization include Power Aware Multi-Access with Signaling (PAMAS) (Stojmenović, 2005); (Karl & Willig, 2005), S-MAC (Ye et al., 2002), Timeout-MAC (T-MAC) (Van Dam & Langendoen, 2003), and B-MAC.

PAMAS is a MAC layer protocol for WSNs that adjusts the duty cycle to minimize radio on time and optimize power consumption. PAMAS uses separate data and control channels (the control channel manages the request/clear to send (RTS/CTS) signals or the receiver busy tone). If a sensor node is receiving a message on the data channel and receives an RTS message on the signaling channel, then the sensor node responds with a busy tone on the signaling channel. This mechanism avoids collisions and results in energy savings. The PAMAS protocol powers off the receiver if either the transmit message queue is empty and the node's neighbor is transmitting or the transmit message queue is not empty but at least one neighbor is transmitting and one neighbor is receiving. WSN simulations with 10 to 20 sensor nodes with 512-byte data packets, 32-byte RTS/CTS packets, and 64-byte busy tone signal packets revealed power savings between 10% and 70% (Singh & Raghavendra, 1998). PAMAS optimization challenges include implementation complexity and associated area cost because the separate control channel requires a second transceiver and duplexer.

The S-MAC protocol tunes the duty cycle and message size for energy conservation. S-MAC minimizes wasted energy due to *frame* (packet) collisions (since collided frames must be retransmitted with additional energy cost), overhearing (a sensor node receiving/listening to a frame destined for another node), control frame overhead, and idle listening (channel monitoring to identify possible incoming messages destined for that node). S-MAC uses a periodic sleep and listen (sleep-sense) strategy defined by the duty cycle. S-MAC avoids frame collisions by using virtual sense (network allocation vector (NAV)-based) and physical carrier sense (receiver listening to the channel) similar to IEEE 802.11. S-MAC avoids overhearing by instructing interfering sensor nodes to switch to sleep mode after hearing an RTS or CTS

packet (Stojmenović, 2005). Experiments conducted on Rene Motes (Culler et al., 2002) for a traffic load comprising of sent messages every 1-10 seconds revealed that a IEEE 802.11-based MAC consumed 2x to 6x more energy than S-MAC (Ye et al., 2004).
T-MAC adjusts the duty cycle dynamically for power efficient operation. T-MAC allows a variable sleep-sense duty cycle as opposed to the fixed duty cycle used in S-MAC (e.g., 10% sense and 90% sleep). The dynamic duty cycle further reduces the idle listening period. The sensor node switches to sleep mode when there is no activation event (e.g., data reception, timer expiration, communication activity sensing, or impending data reception knowledge through neighbors' RTS/CTS) for a predetermined period of time. Experimental results obtained from T-MAC protocol implementation on OMNeT++ (Varga, 2001) to model EYES sensor nodes (EYES, 2010) revealed that under homogeneous load (sensor nodes sent packets with 20- to 100-byte payloads to their neighbors at random), both T-MAC and S-MAC yielded 98% energy savings as compared to CSMA whereas T-MAC outperformed S-MAC by 5x under variable load (Raghavendra et al., 2004).
B-MAC adjusts the duty cycle for power conservation using channel assessment information. B-MAC duty cycles the radio through a periodic channel sampling mechanism known as low power listening (LPL). Each time a sensor node wakes up, the sensor node turns on the radio and checks for channel activity. If the sensor node detects activity, the sensor node powers up and stays awake for the time required to receive an incoming packet. If no packet is received, indicating inaccurate activity detection, a time out forces the sensor node to sleep mode. B-MAC requires an accurate clear channel assessment to achieve low power operation. Experimental results obtained from B-MAC and S-MAC implementation on TinyOS using Mica2 motes revealed that B-MAC power consumption was within 25% of S-MAC for low throughputs (below 45 bits per second) whereas B-MAC outperformed S-MAC by 60% for higher throughputs. Results indicated that B-MAC performed better than S-MAC for latencies under 6 seconds whereas S-MAC yielded lower power consumption as latency approached 10 seconds (Polastre et al., 2004).

5. Network-level Data Dissemination and Routing Protocol Optimizations

One commonality across diverse WSN application domains is the sensor node's task to sense and collect data about a phenomenon and transmit the data to the sink node. To meet application requirements, this data dissemination requires energy-efficient routing protocols to establish communication paths between the sensor nodes and the sink. Typically harsh operating environments coupled with stringent resource and energy constraints make data dissemination and routing challenging for WSNs. Ideally, data dissemination and routing protocols should target energy efficiency, robustness, and scalability. To achieve these optimization objectives, routing protocols adjust transmission power, routing strategies, and leverage either single-hop or multi-hop routing. In this section, we discuss protocols, which optimize data dissemination and routing in WSNs.

5.1 Query Dissemination Optimizations

Query dissemination (transmission of a sensed data query/request from a sink node to a sensor node) and data forwarding (transmission of sensed data from a sensor node to a sink node) requires routing layer optimizations. Protocols that optimize query dissemination and data forwarding include Declarative Routing Protocol (DRP) (Coffin et al., 2000), directed diffusion (Intanagonwiwat et al., 2003), GRAdient Routing (GRAd) (Poor, 2010), GRAdient

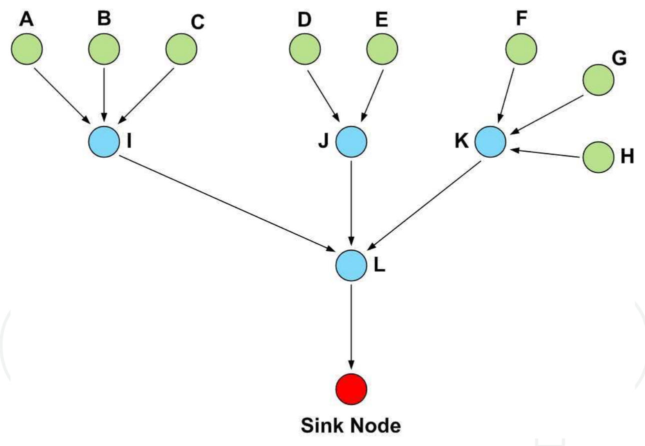


Fig. 3. Data aggregation.

Broadcast (GRAB) (Ye et al., 2005), and Energy Aware Routing (EAR) (Raghavendra et al., 2004); (Shah & Rabaey, 2002).

DRP targets energy efficiency by exploiting in-network aggregation (multiple data items are aggregated as they are forwarded by sensor nodes). Fig. 3 shows in-network data aggregation where sensor node I aggregates sensed data from source nodes A, B, and C, sensor node J aggregates sensed data from source nodes D and E, and sensor node K aggregates sensed data from source nodes F, G, and H. The sensor node L aggregates the sensed data from sensor nodes I, J, and K, and transmits the aggregated data to the sink node. DRP uses reverse path forwarding where data reports (packets containing sensed data in response to query) flow in the reverse direction of the query propagation to reach the sink.

Directed diffusion targets energy efficiency, scalability, and robustness under network dynamics using reverse path forwarding. Directed diffusion builds a shared mesh to deliver data from multiple sources to multiple sinks. The sink node disseminates the query, a process referred to as *interest propagation* (Fig. 4(a)). When a sensor node receives a query from a neighboring node, the sensor node sets up a vector called the *gradient* from itself to the neighboring node and directs future data flows on this gradient (Fig. 4(b)). The sink node receives an initial batch of data reports along multiple paths and uses a mechanism called *reinforcement* to select a path with the best forwarding quality (Fig. 4(c)). To handle network dynamics such as sensor node failures, each data source floods data reports periodically at lower rates to maintain alternate paths. Directed diffusion challenges include formation of initial gradients and wasted energy due to redundant data flows to maintain alternate paths.

GRAd optimizes data forwarding and uses cost-field based forwarding where the cost metric is based on the hop count (i.e., sensor nodes closer to the sink node have smaller costs and those farther away have higher costs). The sink node floods a REQUEST message and the data source broadcasts the data report containing the requested sensed information. The neighbors with smaller costs forward the report to the sink node. GRAd drawbacks include wasted energy due to redundant data report copies reaching the sink node.

GRAB optimizes data forwarding and uses cost-field based forwarding where the cost metric denotes the total energy required to send a packet to the sink node. GRAB was designed for harsh environments with high channel error rate and frequent sensor node failures. GRAB controls redundancy by controlling the width (number of routes from the source sensor node

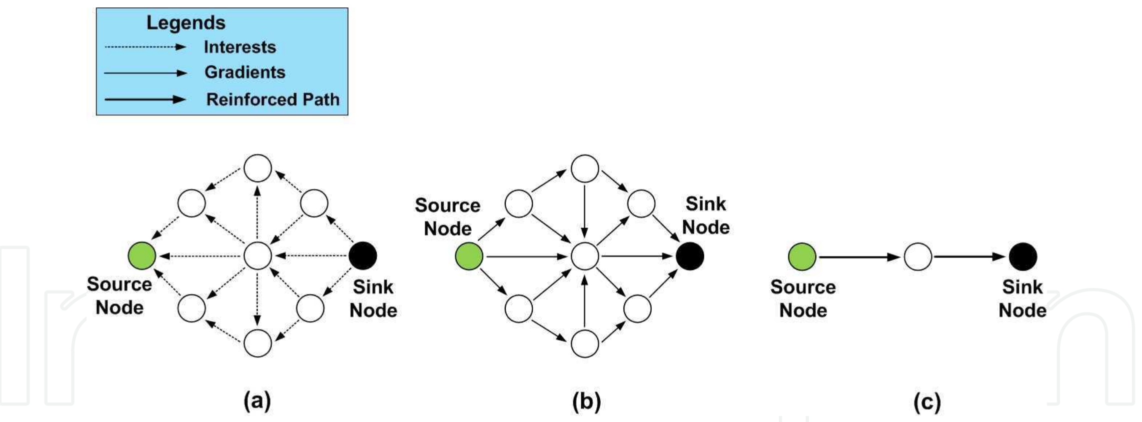


Fig. 4. Directed diffusion: (a) Interest propagation; (b) Initial gradient setup; (c) Data delivery along the reinforced path.

to the sink node) of the forwarding mesh but requires that sensor nodes make assumptions about the energy required to transmit a data report to a neighboring node.

EAR optimizes data forwarding and uses cost-field based forwarding where the cost metric denotes energy per neighbor. EAR optimization objectives are load balancing and energy conservation. EAR makes forwarding decisions probabilistically where the assigned probability is inversely proportional to the neighbor energy cost so that paths consuming more energy are used less frequently (Raghavendra et al., 2004).

5.2 Real-Time Constrained Optimizations

Critical WSN applications may have real-time requirements for sensed data delivery (e.g., a security/defense system monitoring enemy troops or a forest fire detection application). Failure to meet the real-time deadlines for these applications can have catastrophic consequences. Routing protocols that consider the timing constraints for real-time requirements include Real-time Architecture and Protocol (RAP) (Lu et al., 2002) and a stateless protocol for real-time communication in sensor networks (SPEED) (He et al., 2003).

RAP provides real-time data delivery by considering the data report expiration time (time after which the data is of little or no use) and the remaining distance the data report needs to travel to reach the sink node. RAP calculates the desired velocity $v = d/t$ where d and t denote the destination distance and packet lifetime, respectively. The desired velocity is updated at each hop to reflect the data report's urgency. A sensor node uses multiple first-in-first-out (FIFO) queues where each queue accepts reports of velocities within a certain range and then schedules transmissions according to a report's degree of urgency (Raghavendra et al., 2004).

SPEED provides real-time data delivery and uses an exponentially weighted moving average for delay calculation. Given a data report with velocity v, SPEED calculates the speed v_i of the report if the neighbor N_i is selected as the next hop and then selects a neighbor with $v_i > v$ to forward the report to (Raghavendra et al., 2004).

5.3 Network Topology Optimizations

Routing protocols can adjust radio transmission power to control network topology (based on routing paths). Low-Energy Adaptive Clustering Hierarchy (LEACH) (Heinzelman et al., 2000) optimizes the network topology for reduced energy consumption by adjusting the radio's transmission power. LEACH uses a hybrid single-hop and multi-hop communication paradigm. The sensor nodes use multi-hop communication to transmit data reports to a cluster head (LEACH determines the cluster head using a randomized distributed algorithm). The cluster head forwards data to the sink node using long-range radio transmission.

5.4 Resource Adaptive Optimizations

Routing protocols can adapt routing activities in accordance with available resources. Sensor Protocols for Information via Negotiation (SPIN) (Kulik et al., 2002) optimizes performance efficiency by using data negotiation and resource adaptation. In data negotiation, sensor nodes associate metadata with nodes and exchange this metadata before actual data transmission begins. The sensor nodes interested in the data content, based on metadata, request the actual data. This data negotiation ensures that data is sent only to interested nodes. SPIN allows sensor nodes to adjust routing activities according to available energy resources. At low energy levels, sensor nodes reduce or eliminate certain activities (e.g., forwarding of metadata and data packets) (Sohraby et al., 2007).

6. Operating System-level Optimizations

A sensor node's operating system (OS) presents optimization challenges because sensor node operation falls between single-application devices that typically do not need an OS and general-purpose devices with resources to run traditional embedded OSs. A sensor node's OS manages processor, radio, I/O buses, and Flash memory, and provides hardware abstraction to application software, task coordination, power management, and networking services. In this section, we discuss several optimizations provided by existing OSs for sensor nodes (Sohraby et al., 2007).

6.1 Event-Driven Optimizations

Sensor nodes respond to events by controlling sensing and actuation activity. Since sensor nodes are event-driven, it is important to optimize the OS for event handling. WSN OSs optimized for event handling include TinyOS (TinyOS, 2010) and PicOS (Akhmetshina et al., 2002).

TinyOS operates using an event-driven model (tasks are executed based on events). TinyOS is written in the nesC programming language and allows application software to access hardware directly. TinyOS's advantages include simple OS code, energy efficiency, and a small memory foot print. TinyOS challenges include introduced complexity in application development and porting of existing C code to TinyOS.

PicOS is an event-driven OS written in C and designed for limited memory microcontrollers. PicOS tasks are structured as a finite state machine (FSM) and state transitions are triggered by events. PicOS is effective for reactive applications whose primary role is to react to events. PicOS supports multitasking and has small memory requirements but is not suitable for real-time applications.

6.2 Dynamic Power Management

A sensor node's OS can control hardware components to optimize power consumption. Examples include Operating System-directed Power Management (OSPM) (Sinha & Chandrakasan, 2001) and MagnetOS (Barr & et al., 2002), each of which provide mechanisms for dynamic power management. OSPM offers greedy-based dynamic power management, which switches the sensor node to a sleep state when idle. Sleep states provide energy conservation, however, transition to sleep state has the overhead of storing the processor state and requires a finite amount of wakeup time. OSPM greedy-based adaptive sleep mechanism disadvantages include wake up delay and potentially missing events during sleep time. MagnetOS provides two online power-aware algorithms and an adaptive mechanism for applications to effectively utilize the sensor node's resources.

6.3 Fault-Tolerance

Since maintenance and repair of sensor nodes is typically not feasible after deployment, sensor nodes require fault-tolerant mechanisms for reliable operation. MANTIS (Abrach & et al., 2003) is a multithreaded OS that provides fault-tolerant isolation between applications by not allowing a blocking task to prevent the execution of other tasks. In the absence of fault-tolerant isolation, if one task executes a conditional loop whose logical condition is never satisfied, then that task will execute in an infinite loop blocking all other tasks. MANTIS facilitates simple application development and allows dynamic reprogramming to update the sensor node's binary code. MANTIS offers a multimodal prototyping environment for testing WSN applications by providing a remote shell and command server to enable inspection of the sensor node's memory and status remotely. MANTIS challenges include context switch time, stack memory overhead (since each thread requires one stack), and high energy consumption.

7. Dynamic Optimizations

Dynamic optimizations enable in-situ parameter tuning and empowers the sensor node to adapt to changing application requirements and environmental stimuli throughout the sensor node's lifetime. Dynamic optimizations are important because application requirements change over time and environmental stimuli/conditions may not be accurately predicted at design time. Although some OS, MAC layer, and routing optimizations discussed in prior sections of this chapter are dynamic in nature, in this section we present additional dynamic optimization techniques for WSNs.

7.1 Dynamic Voltage and Frequency Scaling

Dynamic voltage and frequency scaling (DVFS) adjusts a sensor node's processor voltage and frequency to optimize energy consumption. DVFS trades off performance for reduced energy consumption by considering that the peak computation (instruction execution) rate is much higher than the application's average throughput requirement and that sensor nodes are based on CMOS logic, which has a voltage dependent maximum operating frequency. Min et al. (Min et al., 2000) demonstrated that a DVFS system containing a voltage scheduler running in tandem with the operating system's task scheduler resulted in a 60% reduction in energy consumption. Yuan et al. (Yuan & Qu, 2002) studied a DVFS system for sensor nodes that required the sensor nodes to insert additional information (e.g., packet length, expected processing time, and deadline) into the data packet's header. The receiving sensor node utilized this information to select an appropriate processor voltage and frequency to minimize the overall energy consumption.

7.2 Software-based Dynamic Optimizations

Software can provide dynamic optimizations using techniques such as duty cycling, batching, hierarchy, and redundancy reduction. Software can control the *duty cycle* so that sensor nodes are powered in a cyclic manner to reduce the average power draw. In *batching*, multiple operations are buffered and then executed in a burst to reduce startup overhead cost. Software can arrange operations in a *hierarchy* based on energy consumption and then invoke low energy operations before high energy operations. Software can reduce *redundancy* by compression, data aggregation, and/or message suppression. Kogekar et al. (Kogekar et al., 2004) proposed an approach for software reconfiguration in WSNs. The authors modeled the WSN operation space (defined by the WSN software components' models and application requirements) and defined reconfiguration as the process of switching from one point in the operation space to another.

7.3 Dynamic Network Reprogramming

Dynamic network reprogramming reprograms sensor nodes to change/modify tasks by disseminating code in accordance with changing environmental stimuli. Since recollection and reprogramming is not a feasible option for most sensor nodes, dynamic network reprogramming enables the sensor nodes to perform different tasks. For example, a WSN initially deployed for measuring relative humidity can measure temperature statistics after dynamic reprogramming. The MANTIS OS provides this dynamic reprogramming ability (Section 6.3).

8. MDP-based Dynamic Optimizations

In this section, we extend our discussion of dynamic optimizations using an MDP-based dynamic optimization (Munir & Gordon-Ross, 2009) as a specific example. MDP is suitable for WSN dynamic optimizations because of MDP's inherent ability to perform dynamic decision making. We propose MDP as a method to perform parameter tuning-based dynamic optimizations. Traditional microprocessor-based systems use DVFS for energy optimizations. DVFS only provides a partial tuning for sensor nodes because sensor nodes are distinct from traditional systems in that they have embedded sensors coupled with an embedded processor. For example, the sensing frequency dictates the amount of processed and communicated data. We propose dynamic voltage, frequency, and sensing frequency scaling (DVFS2) to provide enhanced optimization potential as compared to DVFS for WSNs. Our MDP-based optimization focuses on DVFS2 but is equally applicable for extensive design spaces with more tunable parameters (e.g., transmission power, packet transmission interval, etc.).

8.1 Dynamic Optimization Methodology

Fig. 5 depicts the process diagram for our dynamic optimization, which consists of three logical domains: the application characterization domain, the communication domain, and the sensor node tuning domain.

The *application characterization domain* refers to the WSN application's characterization/specification where the application manager/designer (one who manages/designs a WSN) defines various *application metrics* (e.g., lifetime, throughput, reliability, etc.) based on application requirements. The application manager/designer also assigns *weight factors* to application metrics which signify the weightage or relative importance of each application metric with respect to other metrics. The *objective function* or *reward function* signifies the overall reward (revenue) for given application requirements. The application metrics along with associated weight factors represent the *objective/reward function parameters*.

The *communication domain* (depicted by the sink node in Fig. 5) encompasses the communication network between the application manager and the sensor nodes. The application manager transmits the objective or reward function parameters to the sink node via the communication domain which in turn relays these parameters to the sensor nodes.

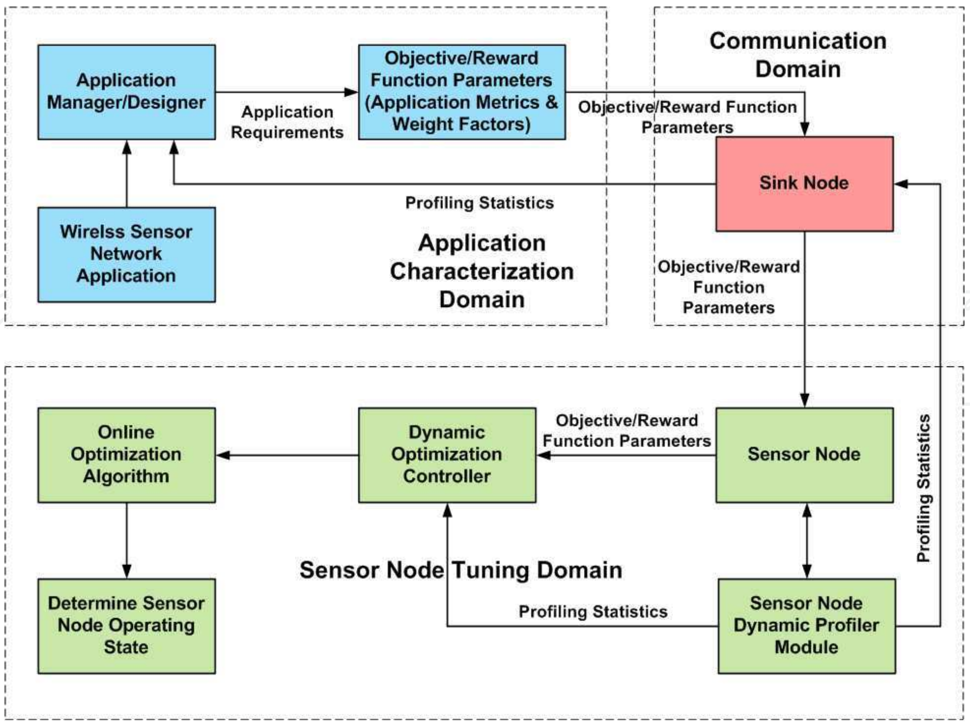

Application Manager/Designer
Application Requirements
Objective/Reward Function Parameters (Application Metrics & Weight Factors)
Communication Domain
Objective/Reward Function Parameters
Sink Node
Wirelss Sensor Network Application
Profiling Statistics
Application Characterization Domain
Objective/Reward Function Parameters
Online Optimization Algorithm
Dynamic Optimization Controller
Objective/Reward Function Parameters
Sensor Node
Profiling Statistics
Sensor Node Tuning Domain
Determine Sensor Node Operating State
Profiling Statistics
Sensor Node Dynamic Profiler Module

Fig. 5. Process diagram for parameter tuning-based dynamic optimizations for WSNs.

The *sensor node tuning domain* consists of sensor nodes and performs sensor node parameter tuning. Each sensor node contains a *dynamic optimization controller*, which orchestrates the dynamic optimization process. The dynamic optimization controller module receives the reward function parameters and invokes an online optimization algorithm to determine an optimal or near-optimal sensor node *state* (tunable parameter value settings).
Our proposed methodology reacts to environmental stimuli via a *dynamic profiler module*, which monitors environmental changes over time and captures unanticipated environmental situations not predictable at design time. The dynamic profiler module profiles the *profiling statistics* (e.g., wireless channel condition, number of packets dropped, battery energy, etc.). The dynamic profiler module informs the dynamic optimization controller as well as the application manager of the profiled statistics. The dynamic optimization controller processes the profiling statistics to determine if the current operating state meets the application requirements. If the current operating state does not meet the application requirements, the dynamic optimization controller reinvokes the online optimization algorithm (e.g., MDP-based or any other) to determine the new operating state. This feedback process continues to ensure the selection of a good operating state to better meet application requirements in the presence of changing environmental stimuli.

8.2 Dynamic Optimization Formulation

In this subsection, we formulate the constructs of our MDP-based dynamic optimization (Munir & Gordon-Ross, 2009). Although we describe dynamic optimization constructs with reference to MDP, our formulation provides insight into any other dynamic optimization algorithm.

8.2.1 State Space

The state space S for our MDP-based dynamic optimization methodology given N tunable parameters is defined as:

$$S = S_1 \times S_2 \times \cdots \times S_N \quad : \quad |S| = I \tag{1}$$

where S_i denotes the state space for tunable parameter i, $\forall \; i \in \{1, 2, \ldots, N\}$ and $\times$ denotes the Cartesian product. The state space S consists of a total of I states as given by the state space cardinality $|S| = I$. Each tunable parameter's state space S_i consists of n tunable values:

$$S_i = \{s_{i_1}, s_{i_2}, s_{i_3}, \ldots, s_{i_n}\} \quad : \quad |S_i| = n \tag{2}$$

where $|S_i|$ denotes the number of tunable values in S_i. S is a set of N-tuples formed by taking one tunable parameter value from each tunable parameter. A single N-tuple $s \in S$ is given as:

$$\begin{aligned} s \;\; = \;\; & (s_{1_y}, s_{2_y}, \ldots, s_{N_y}) : s_{i_y} \in S_i, \\ & \forall \; i \in \{1, 2, \ldots, N\}, \; y \in \{1, 2, \ldots, n\} \end{aligned} \tag{3}$$

Each N-tuple represents a sensor note state. We point out that some N-tuples in S may not be feasible (such as invalid combinations of processor voltage and frequency) and can be regarded as *do not care* tuples.
For example, given three tunable parameters, S can be written as:

$$S = V_p \times F_p \times F_s \tag{4}$$

where V_p, F_p, and F_s denote the state space for a sensor node's processor voltage, processor frequency, and sensing (sampling) frequency, respectively.

8.2.2 Decision Epochs and Actions

The *decision epochs* refer to the points of time during a sensor node's lifetime at which the sensor node makes a decision regarding its operating state (i.e., whether to continue operating in the current state or transition to another state). We consider a discrete time process where time is divided into *periods* and a decision epoch corresponds to the beginning of a period. The sequence of decision epochs is represented as:

$$T = \{1, 2, 3, ..., N\}, \quad N \leq \infty \tag{5}$$

where the random variable N corresponds to the sensor node's lifetime (each individual time period in T can be denoted as time t).
At each decision epoch, a sensor node's *action* determines the next state to transition to given the current state. The sensor node action in state $i \in S$ is defined as:

$$A_i = \{a_{i,j}\} \in \{0, 1\} \tag{6}$$

where $a_{i,j}$ denotes the action taken at time t that causes the sensor node to transition to state j at time $t + 1$ from the current state i. If $a_{i,j} = 1$, the action is taken and if $a_{i,j} = 0$, the action is not taken.

8.2.3 Policy and Performance Criterion

For each given state $s \in S$, a *policy* π determines whether an action $a \in A_s$ is taken or not at a decision epoch. A *performance criterion* compares the performance of different policies. The sensor node selects an action prescribed by a policy based on the sensor node's current state. The sensor node receives a reward $r\left(X_t, Y_t\right)$ as a result of selecting an action Y_t at decision epoch t where the random variable X_t denotes the state at decision epoch t. The *expected total reward* $v_N^{\pi}(s)$ denotes the expected total reward over the decision making horizon N given a specific policy π (Puterman, 2005); (Stevens-Navarro et al., 2008):

$$v_N^{\pi}(s) = \lim_{N \to \infty} E_s^{\pi}\left[E_N\left\{\sum_{t=1}^{N} r(X_t, Y_t)\right\}\right] \tag{7}$$

where E_s^{π} represents the expected reward with respect to policy π and the initial state s (the system state at the time of the expected reward calculation) and E_N denotes the expected reward with respect to the probability distribution of the random variable N. We can write (7) as (Puterman, 2005):

$$v_N^{\lambda}(s) = E_s^{\pi}\left\{\sum_{t=1}^{\infty} \lambda^{t-1} r(X_t, Y_t)\right\} \tag{8}$$

which gives the *expected total discounted reward*. We assume that the random variable N is geometrically distributed with parameter λ and hence the distribution *mean* is $1/(1-\lambda)$ (Stevens-Navarro et al., 2008). The parameter λ can be interpreted as a *discount factor*, which measures the present value of one unit of reward received one period in the future. Thus, $v_N^{\lambda}(s)$ represents the expected total present value of the reward (income) stream obtained using policy π (Puterman, 2005). Our objective is to find a policy that maximizes the expected total discounted reward i.e., a policy π^* is *optimal* if:

$$v^{\pi^*}(s) \geq v^{\pi}(s) \quad \forall \;\; \pi \in \Pi \tag{9}$$

where Π denotes the set of admissible policies.

8.2.4 State Dynamics

The state dynamics of the system (sensor node) can be delineated by the state transition probabilities of the embedded Markov chain. We formulate our sensor node policy as a deterministic dynamic program (DDP) because the choice of an action determines the subsequent state with certainty. Our sensor node DDP policy formulation uses a *transfer function* to specify the next state. A transfer function defines a mapping $\tau_t(s,a)$ from $S \times A_s \to S$, which specifies the system state at time $t+1$ when the sensor node selects action $a \in A_s$ in state s at time t. To formulate our DDP as an MDP, we define the *transition probability function* as:

$$p_t(j|s,a) = \begin{cases} 1 & \text{if } \;\tau_t(s,a) = j \\ 0 & \text{if } \;\tau_t(s,a) \neq j \end{cases} \tag{10}$$

8.2.5 Reward Function

The reward function captures application metrics and sensor node characteristics. Our reward function characterization considers the power consumption (which affects the sensor node

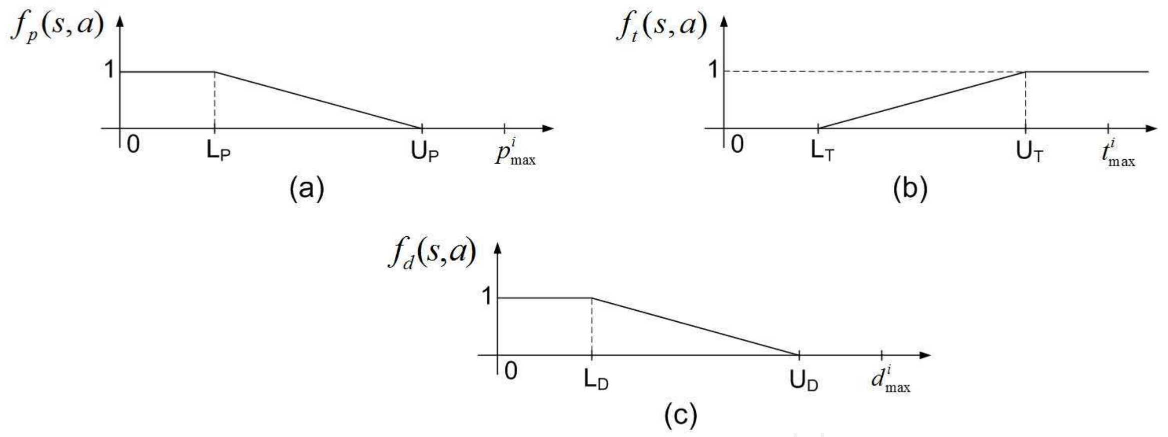


Fig. 6. Reward functions: (a) Power reward function $f_p(s,a)$; (b) Throughput reward function $f_t(s,a)$; (c) Delay reward function $f_d(s,a)$.

lifetime), throughput, and delay application metrics. We define the reward function $f(s,a)$ given the current sensor node state s and the sensor node's selected action a as:

$$\begin{aligned} f(s,a) &= \sum_{k=1}^{m} \omega_k f_k(s,a) \\ s.t. \quad & s \in S \\ & \omega_k \geq 0, \quad k = 1,2,\ldots,m \\ & \omega_k \leq 1, \quad k = 1,2,\ldots,m \\ & \sum_{k=1}^{m} \omega_k = 1, \end{aligned} \tag{11}$$

where $f_k(s,a)$ and ω_k denote the reward function and weight factor for the k^{th} application metric, respectively, given that there are m application metrics. Our objective function characterization considers power, throughput, and delay (i.e., $m = 3$) (additional application metrics can be included) and is given as:

$$f(s,a) = \omega_p f_p(s,a) + \omega_t f_t(s,a) + \omega_d f_d(s,a) \tag{12}$$

where $f_p(s,a)$ denotes the power reward function, $f_t(s,a)$ denotes the throughput reward function, and $f_d(s,a)$ denotes the delay reward function (Fig. 6); ω_p, ω_t, and ω_d represent the *weight factors* for power, throughput, and delay, respectively.

We define linear reward functions for application metrics because an application metric reward (objective function) typically varies linearly, or piecewise linearly, between the minimum and maximum allowed values of the metric (Stevens-Navarro et al., 2008). However, a non-linear characterization of reward functions is also possible and depends upon the particular application. Our methodology works for any characterization of reward function. We define the power reward function (Fig. 6(a)) in (11) as:

$$f_p(s,a) = \begin{cases} 1, & 0 < p_a \leq L_P \\ (U_P - p_a)/(U_P - L_P), & L_P < p_a < U_P \\ 0, & p_a \geq U_P \end{cases} \tag{13}$$

where p_a denotes the power consumption of the current state given action a taken at time t and the constant parameters L_P and U_P denote the minimum and maximum allowed/tolerated sensor node power consumption, respectively. Similar equations can be written for $f_t(s,a)$ and $f_d(s,a)$.
State transitioning incurs a cost associated with switching parameter values from the current state to the next state (typically in the form of power and/or execution (time) overhead). We define the transition cost function $h(s,a)$ as:

$$h(s,a) = \begin{cases} H_{i,a} & \text{if } i \neq a \\ 0 & \text{if } i = a \end{cases} \tag{14}$$

where $H_{i,a}$ denotes the transition cost to switch from the current state i to the next state as determined by action a. Note that a sensor node incurs no transition cost if action a prescribes that the next state is the same as the current state.
Hence, the overall reward function $r(s,a)$ given state s and action a at time t is:

$$r(s,a) = f(s,a) - h(s,a) \tag{15}$$

which accounts for the power, throughput, and delay application metrics as well as state transition cost.

8.2.6 Optimality Equation

The optimality equation, also known as Bellman's equation, for expected total discounted reward criterion is given as (Puterman, 2005):

$$v(s) = \max_{a \in A_s} \left\{ r(s,a) + \sum_{j \in S} \lambda p(j|s,a) v(j) \right\} \tag{16}$$

where $v(s)$ denotes the maximum expected total discounted reward. The salient properties of the optimality equation are: the optimality equation has a unique solution; an optimal policy exists given conditions on states, actions, rewards, and transition probabilities; the value of the discounted MDP satisfies the optimality equation; and the optimality equation characterizes stationary policies.
The solution of (16) gives the maximum expected total discounted reward $v(s)$ and the MDP-based optimal policy π^* (or π^{MDP}), which gives the maximum $v(s)$. π^{MDP} prescribes the action a from action set A_s given the current state s for all $s \in S$. There are several methods to solve the optimality equation (16) such as value iteration, policy iteration, and linear programming, however in this work we use the policy iteration algorithm. The details of the policy iteration algorithm can be found in (Puterman, 2005).

8.3 Numerical Results

In this section, we compare the performance (based on expected total discounted reward criterion) of our proposed MDP-based DVFS2 optimal policy π^* (π^{MDP}) with several fixed heuristic policies using a representative WSN platform. We use the MATLAB MDP tool box (Chadès et al., 2005) implementation of the policy iteration algorithm (Puterman, 2005) to determine the MDP-based optimal policy. Given the reward function, sensor node state parameters, and transition probabilities, (8) gives the expected total discounted reward. Our reference WSN platform consists of eXtreme Scale Motes (XSM) sensor nodes (Dutta et al.,

Parameter	$i_1 = [2.7, 2, 2]$	$i_2 = [3, 4, 4]$	$i_3 = [4, 6, 6]$	$i_4 = [5.5, 8, 8]$
p_i	10 units	15 units	30 units	55 units
t_i	4 units	8 units	12 units	16 units
d_i	26 units	14 units	8 units	6 units

Table 2. Power consumption p_i, throughput t_i, and delay d_i parameters for wireless sensor node state $i = [V_p, F_p, F_s]$ (V_p is specified in volts, F_p in MHz, and F_s in KHz). Parameters are specified as a multiple of a base unit where one power unit is equal to 1 mW, one throughput unit is equal to 0.5 MIPS, and one delay unit is equal to 50 ms. Parameter values are based on the XSM mote.

2005); (Dutta & Culler, 2005). The XSM motes have an average lifetime of 1,000 hours of continuous operation with two AA alkaline batteries, which can deliver 6 Whr or an average of 6 mW (Dutta et al., 2005). The XSM platform integrates an Atmel ATmega128L microcontroller (ATMEL, 2009), a Chipcon CC1000 radio operating at 433 MHz, and a 4 Mbit serial flash memory. The XSM motes contain infra red, magnetic, acoustic, photo, and temperature sensors. To represent sensor node operation, we analyze a sample application domain that represents a typical security system or defense application (henceforth referred to as a *security/defense system*).

8.3.1 Fixed Heuristic Policies for Performance Comparisons

We consider the following four fixed heuristic policies for comparison with our MDP policy:

- A fixed heuristic policy π^{POW} that always selects the state with the lowest power consumption.
- A fixed heuristic policy π^{THP} that always selects the state with the highest throughput.
- A fixed heuristic policy π^{EQU} that spends an equal amount of time in each of the available states.
- A fixed heuristic policy π^{PRF} that spends an unequal amount of time in each of the available states based on a specified preference for each state. For example, given a system with four possible states, the π^{PRF} policy may spend 40% of the time in the first state, 20% of the time in the second state, 10% of the time in the third state, and 30% of the time in the fourth state.

8.3.2 MDP Specifications

We compare different policies using the *expected total discounted reward* performance criterion. The state transition probability for each sensor node state is given by (10). The sensor node's lifetime and the time between decision epochs are subjective and may be assigned by an application manager according to application requirements. A sensor node's *mean lifetime* is given by $1/(1-\lambda)$ *time units*, which is the time between successive decision epochs (which we assume to be 1 hour). For instance for $\lambda = 0.999$, the sensor node's mean lifetime is $1/(1-0.999) = 1000$ hours ≈ 42 days.

For our numerical results, we consider a sensor node capable of operating in four different states (i.e., $I = 4$ in (1)). Each state has a set of allowed actions prescribing transitions to available states. For each allowed action a in a state, there is a $\{r_a, p_a\}$ pair where r_a specifies the immediate reward obtained by taking action a and p_a denotes the probability of taking action a.

λ	**Sensor Lifetime**	π^{MDP}	π^{POW}	π^{THP}	π^{EQU}	π^{PRF}
0.94	16.67 hours	10.0006	7.5111	9.0778	7.2692	7.5586
0.95	20 hours	12.0302	9.0111	10.9111	8.723	9.0687
0.96	25 hours	15.0747	11.2611	13.6611	10.9038	11.3339
0.97	33.33 hours	20.1489	15.0111	18.2445	14.5383	15.1091
0.98	50 hours	30.2972	22.5111	27.4111	21.8075	22.6596
0.99	100 hours	60.7422	45.0111	54.9111	43.6150	45.3111
0.999	1000 hours	608.7522	450.0111	549.9111	436.15	453.0381
0.9999	10,000 hours	6088.9	4500	5499.9	4361.5	4530.3
0.99999	100,000 hours	60890	45000	55000	43615	45303

Table 4. The effects of different discount factors λ for a security/defense system. $H_{i,j} = 0.1$ if $i \neq j, \omega_p = 0.45, \omega_t = 0.2, \omega_d = 0.35$.

Table 2 summarizes state parameter values for each of the four states i_1, i_2, i_3, and i_4. We define each state using a $[V_p, F_p, F_s]$ tuple where V_p is specified in volts, F_p in MHz, and F_s in KHz. For instance, state one i_1 is defined as $[2.7, 2, 2]$, which corresponds to a processor voltage of 2.7 volts, a processor frequency of 2 MHz, and a sensing frequency of 2 KHz (2000 samples per second). We assume, without loss of generality, that the transition cost for switching from one state to another is $H_{i,a} = 0.1$ if $i \neq a$.

Our selection of the state parameter values in Table 2 corresponds to XSM mote specifications. The XSM mote's Atmel ATmega128L microprocessor has an operating voltage range of 2.7 to 5.5 V and a processor frequency range of 0 to 8 MHz. The ATmega128L throughput varies with processor frequency at 1 MIPS per MHz, thus allowing a WSN designer to optimize power consumption versus processing speed (ATMEL, 2009). Our chosen sensing frequency also corresponds with standard sensor node specifications. The Honeywell HMC1002 magnetometer sensor (Honeywell, 2009) consumes on average 15 mW of power and can be sampled in 0.1 ms on the Atmel ATmega128L microprocessor, which results in a maximum sampling frequency of approximately 10 KHz (10,000 samples per second). The acoustic sensor embedded in the XSM mote has a maximum sensing frequency of approximately 8.192 KHz (Dutta et al., 2005).

Table **??** summarizes the minimum L and maximum U reward function parameter values for application metrics (power, throughput, and delay) and associated weight factors for a security/defense system. We selected reward function parameter values according to typical application requirements for a security/defense system (Akyildiz et al., 2002). For instance, a data sensitive and time critical security/defense system with stringent minimum and maximum tolerable delay might require a comparatively large minimum throughput in order to obtain a sufficient number of sensed data samples for meaningful analysis.

For brevity, we select a single sample WSN platform configuration and application, but we point out that our proposed MDP model and methodology works equally well for any other WSN platform and application.

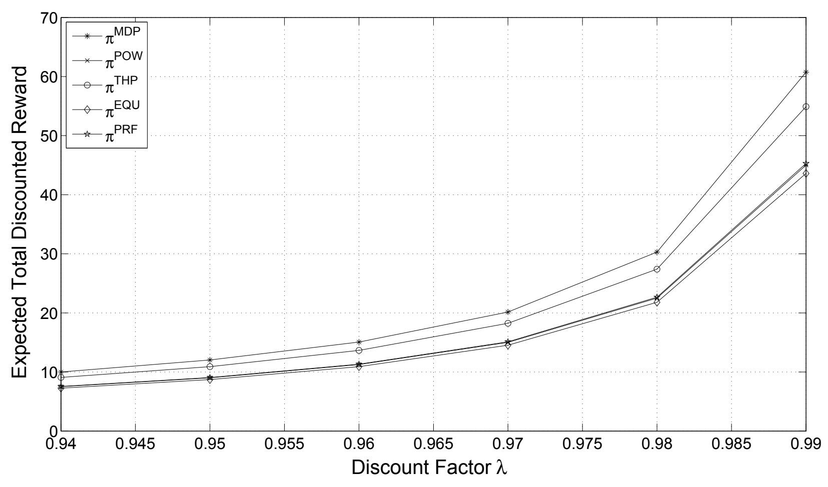


Fig. 7. The effects of different discount factors on the expected total discounted reward for a security/defense system. $H_{i,j} = 0.1$ if $i \neq j, \omega_p = 0.45, \omega_t = 0.2, \omega_d = 0.35$.

8.3.3 Results

In this subsection, we present the results for a security/defense system for our MDP-based optimal policy and the fixed heuristic policies (Section 8.3.1). We evaluate the effects of different discount factors, different state transition costs, and different application metric weight factors on the expected total discounted reward. The magnitude of difference in the total expected discounted reward for different policies is important as it provides relative comparisons between the different policies.

Table 4 and Figure 7 depict the effects of different discount factors λ on the heuristic policies and π^{MDP} for a security/defense system when the state transition cost $H_{i,j}$ is held constant at 0.1 for $i \neq j$. Since we assume the time between successive decision epochs to be 1 hour, the range of λ from 0.94 to 0.99999 corresponds to a range of average sensor node lifetime from 16.67 to 100,000 hours $\approx$ 4167 days $\approx$ 11.4 years. Table 4 and Figure 7 show that π^{MDP} results in the highest expected total discounted reward for all values of λ and corresponding average sensor node lifetimes. For instance, when the average sensor node lifetime is 1,000 hours ($\lambda = 0.999$), π^{MDP} results in a 26.08%, 9.67%, 28.35%, and 25.58% increase in expected total discounted reward as compared to π^{POW}, π^{THP}, π^{EQU}, and π^{PRF}, respectively. On average over all discount factors λ, π^{MDP} results in a 25.57%, 9.48%, 27.91%, and 25.1% increase in expected total discounted reward as compared to π^{POW}, π^{THP}, π^{EQU}, and π^{PRF}, respectively.

Figure 8 depicts the effects of different state transition costs on the expected total discounted reward for a security/defense system with a fixed average sensor node lifetime of 1000 hours ($\lambda = 0.999$). Figure 8 shows that π^{MDP} results in the highest expected total discounted reward for all transition cost values. Figure 8 also shows that the expected total discounted reward for π^{MDP} is relatively unaffected by state transition cost. This relatively constant behavior can be explained by the fact that our MDP optimal policy does not perform many state transitions. π^{MDP} performs state transitions primarily at sensor node deployment or whenever a new MDP-based optimal policy is determined as the result of changes in application requirements.

Figure 9 shows the effects of different reward function weight factors on the expected total discounted reward for a security/defense system when the average sensor node lifetime is 1,000 hours ($\lambda = 0.999$) and the state transition cost $H_{i,j}$ is held constant at 0.1 for $i \neq j$. We explore various weight factors that are appropriate for different security/defense system

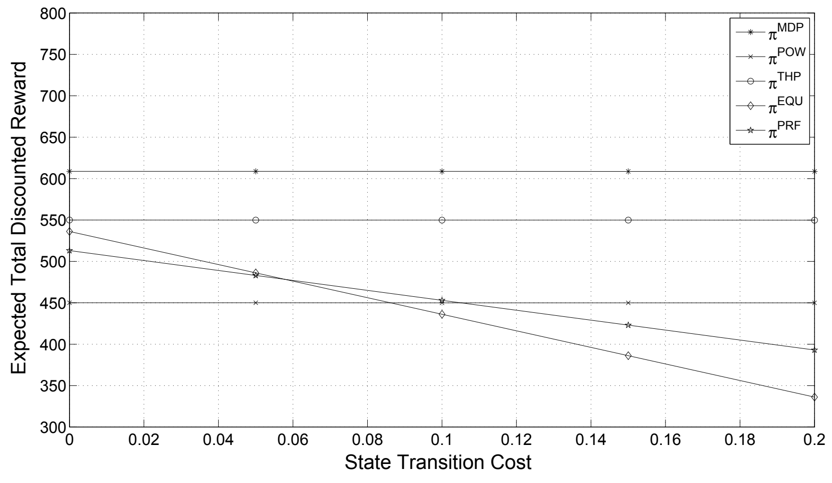


Fig. 8. The effects of different state transition costs on the expected total discounted reward for a security/defense system. $\lambda = 0.999$, $\omega_p = 0.45, \omega_t = 0.2, \omega_d = 0.35$.

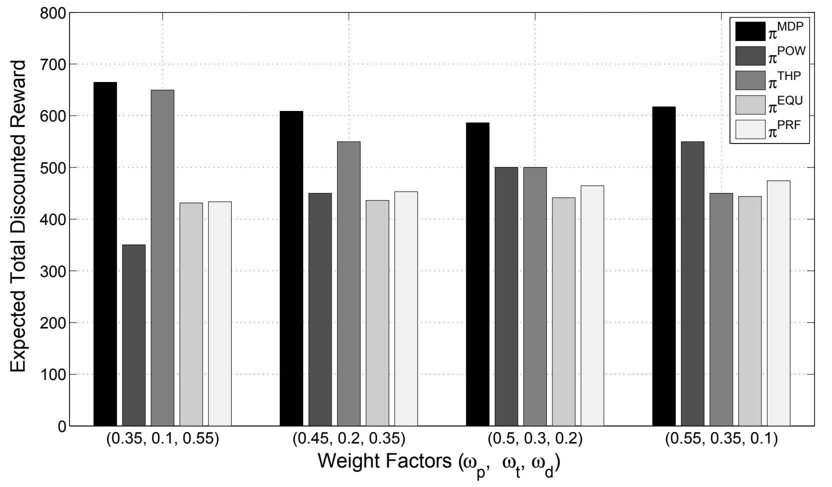


Fig. 9. The effects of different reward function weight factors on the expected total discounted reward for a security/defense system. $\lambda = 0.999$, $H_{i,j} = 0.1$ if $i \neq j$

specifics (i.e., $(\omega_p, \omega_t, \omega_d)$ = {(0.35, 0.1, 0.55), (0.45, 0.2, 0.35), (0.5, 0.3, 0.2), (0.55, 0.35, 0.1)}). Figure 9 reveals that π^{MDP} results in the highest expected total discounted reward for all weight factor variations.

9. Conclusions

WSNs have been employed in diverse application domains each with different and competing application requirements. Given this diversity, meeting application requirements is a challenging design task. Optimization techniques at different design levels help meet these application requirements. In this chapter, we discussed WSNs from an optimization perspective. We presented a typical WSN architecture along with several possible integration scenarios with external IP networks for ubiquitous availability of WSN offered services (e.g., sensed temperature and humidity statistics). We discussed COTS sensor node components

and associated tunable parameters that can be specialized to provide component-level optimizations. We presented data link-level and network-level optimization strategies focusing on MAC and routing protocols, respectively. Our presented MAC protocols targeted load balancing, throughput, and energy optimizations and routing protocols addressed query dissemination, real-time data delivery, and network topology. Different OS optimizations include event-driven execution, dynamic power management, and fault-tolerance.
Even though many of the optimizations offered by MAC, routing, and the OS are dynamic in nature, we focused on dynamic optimizations separately due to their increasing research significance. Traditional DVFS-based optimizations only tune processor voltage and frequency, however, sensor nodes possess other tunable parameters (e.g., sensing frequency, transmission power) whose tuning can increase the potential for meeting application requirements. In this chapter, we proposed an MDP-based dynamic optimization methodology to optimally tune sensor node parameters. Our proposed methodology is adaptive and dynamically determines the new MDP-based optimal policy (sensor node operating state) whenever application requirements change (which may be in accordance with changing environmental stimuli). We compared our MDP-based methodology with four fixed heuristic policies. Numerical results revealed that our MDP-based policy outperformed other heuristic policies for all sensor node lifetimes, state transition costs, and application metric weight factors. Future research trends in WSN dynamic optimizations include the investigation of lightweight online algorithms suitable for sensor nodes with constrained resources and incorporation of profiling statistics to provide feedback to the optimization algorithms.

Acknowledgments

This work was supported by the National Science Foundation (CNS-0834080) and the Natural Sciences and Engineering Research Council of Canada (NSERC). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the NSF or NSERC.

10. References

Abrach, H. & et al. (2003). MANTIS: System Support for Multimodal Networks of In-Situ Sensors, *Proc. of Workshop on Wireless Sensor Networks and Applications (WSNA)'03*, San Diego, California, pp. 50–59.

Abramson, N. (1985). Development of the ALOHANET, *IEEE Trans. on Information Theory* **31**(2): 119–123.

Akhmetshina, E., Gburzynski, P. & Vizeacoumar, F. (2002). PicOS: A Tiny Operating System for Extremely Small Embedded Platforms, *Proc. of Conference on Embedded Systems and Applications (ESA)'02*, Las Vegas, Nevada, pp. 116–122.

Akyildiz, I., Su, W., Sankarasubramaniam, Y. & Cayirci, E. (2002). Wireless Sensor Networks: A Survey, *Elsevier Computer Networks* **38**(4): 393–422.

ATMEL (2009). ATMEL ATmega128L 8-bit Microcontroller Datasheet, *ATMEL Corporation*, San Jose, California.
URL: *http://www.atmel.com/dyn/resources/prod_documents/doc2467.pdf*

Bao, L. & Garcia-Luna-Aceves, J. (2001). A New Approach to Channel Access Scheduling for Ad Hoc Networks, *Proc. of MobiCom'01*, ACM, Rome, Italy.

Barr, R. & et al. (2002). On the Need for System-Level Support for Ad Hoc and Sensor Networks, *ACM SIGOPS Operating Systems Review* **36**(2): 1–5.

Brooks, D. & Martonosi, M. (2000). Value-based Clock Gating and Operation Packing: Dynamic Strategies for Improving Processor Power and Performance, *ACM Trans. on Computer Systems* **18**(2): 89–126.

Chadès, I., Cros, M., Garcia, F. & Sabbadin, R. (2005). Markov Decision Process (MDP) Toolbox v2.0 for MATLAB, *INRA Toulouse*, INRA, France.
URL: *http://www.inra.fr/internet/Departements/MIA/T/MDPtoolbox/*

Chandrakasan, A., Amirtharajah, R., Cho, S., Konduri, J., Kulik, J., Rabiner, W. & Wang, A. (1999). Design Considerations for Distributed Microsensor Systems, *Proc. of IEEE Custom Integrated Circuits Conference (CICC)*, San Diego, California.

Coffin, D., Hook, D., McGarry, S. & Kolek, S. (2000). Declarative Ad-hoc Sensor Networking, *In SPIE Integrated Command Environments*.

Culler, D., Hill, J., Horton, M., Pister, K., Szewczyk, R. & Woo, A. (2002). MICA: The Commercialization of Microsensor Motes, *Sensor Magazine*.
URL: *http://www.sensorsmag.com/articles/0402/40/*

Dutta, P. & Culler, D. (2005). System Software Techniques for Low-Power Operation in Wireless Sensor Networks, *Proc. of IEEE/ACM ICCAD*, San Jose, California.

Dutta, P., Grimmer, M., Arora, A., Bibyk, S. & Culler, D. (2005). Design of a Wireless Sensor Network Platform for Detecting Rare, Random, and Ephemeral Events, *Proc. of ACM IPSN*, Los Angeles, California.

EYES (2010). Energy Efficient Sensor Networks.
URL: *http://www.eyes.eu.org/sensnet.htm*

Hamed, H., El-Atawy, A. & Ehab, A.-S. (2006). On Dynamic Optimization of Packet Matching in High-Speed Firewalls, *IEEE Journal on Selected Areas in Communications* **24**(10): 1817–1830.

Hazelwood, K. & Smith, M. D. (2006). Managing Bounded Code Caches in Dynamic Binary Optimization Systems, *ACM Trans. on Architecture and Code Optimization* **3**(3): 263–294.

He, T., Stankovic, J., Lu, C. & Abdelzaher, T. (2003). SPEED: A Stateless Protocol for Real-time Communication in Sensor Networks, *Proc. of International Conference on Distributed Computing Systems (ICDCS)'03*, IEEE, Providence, Rhode Island.

Heinzelman, W., Chandrakasan, A. & Balakrishnan, H. (2000). Energy-Efficient Communication Protocols for Wireless Microsensor Networks, *Hawaiian International Conference on System Sciences*.

Honeywell (2009). Honeywell 1- and 2- Axis Magenetic Sensors HMC1001/1002, and HMC1021/1022 Datasheet, *Honeywell International Inc.*, Morristown, New Jersey.
URL: *http://www.ssec.honeywell.com/magnetic/datasheets/hmc1001-2_1021-2.pdf*

Hu, S., Valluri, M. & John, L. K. (2006). Effective Management of Multiple Configurable Units using Dynamic Optimization, *ACM Trans. on Architecture and Code Optimization* **3**(4): 477–501.

IEEE Standards (1999). Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification, IEEE Std 802.11-1999 edition: LAN MAN Standards Committee of the IEEE Computer Society.

Intanagonwiwat, C., Govindan, R., Estrin, D., Heidemann, J. & Silva, F. (2003). Directed Diffusion for Wireless Sensor Networking, *IEEE/ACM Trans. on Networking* **11**(1): 2–16.

Karl, H. & Willig, A. (2005). *Protocols and Architectures for Wireless Sensor Networks*, John Wiley and Sons, Inc.

Kogekar, S., Neema, S., Eames, B., Koutsoukos, X., Ledeczi, A. & Maroti, M. (2004). Constraint-Guided Dynamic Reconfiguration in Sensor Networks, *Proc. of ACM IPSN*, Berkeley, California.

Kulik, J., Heinzelman, W. & Balakrishnan, H. (2002). Negotiation-Based Protocols for Disseminating Information in Wireless Sensor Networks, *ACM Wireless Networks (WINET)* **8**(2/3): 169–185.

Lu, C., Blum, B., Abdelzaher, T., Stankovic, J. & He, T. (2002). RAP: A Real-Time Communication Architecture for Large-Scale Wireless Sensor Networks, *Real-Time and Embedded Technology and Applications Symposium (RTAS)'02*, San Jose, California.

Mahalik, N. (2007). *Sensor Networks and Configuration: Fundamentals, Standards, Platforms, and Applications*, Springer.

Min, R., Furrer, T. & Chandrakasan, A. (2000). Dynamic Voltage Scaling Techniques for Distributed Microsensor Networks, *Proc. of IEEE WVLSI*, Orlando, Florida.

Munir, A. & Gordon-Ross, A. (2009). An MDP-based Application Oriented Optimal Policy for Wireless Sensor Networks, *Proc. of International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS'09)*, ACM, Grenoble, France, pp. 183–192.

Polastre, J., Hill, J. & Culler, D. (2004). Versatile Low Power Media Access for Wireless Sensor Networks, *Proc. of International Conference on Embedded Networked Sensor Systems (SenSys)'04*, ACM, Baltimore, Maryland.

Poor, R. (2010). Gradient Routing in Ad Hoc Networks.
URL: *http://www.media.mit.edu/pia/Research/ESP/texts/poorieeepaper.pdf*

Puterman, M. (2005). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, John Wiley and Sons, Inc.

Raghavendra, C., Sivalingam, K. & Znati, T. (2004). *Wireless Sensor Networks*, Kluwer Academic Publishers.

Rajendran, V., Obraczka, K. & Garcia-Luna-Aceves, J. (2003). Energy-efficient Collision-free Medium Access Control for Wireless Sensor Networks, *Proc. of International Conference on Embedded Networked Sensor Systems (SenSys)'03*, ACM, Los Angeles, California.

Rappaport, T. S. (1996). *Wireless Communications, Principles and Practice*, Prentice Hall.

Rhee, I., Warrier, A., Aia, M. & Min, J. (2005). Z-MAC: A Hybrid MAC for Wireless Sensor Networks, *Proc. of International Conference on Embedded Networked Sensor Systems (SenSys)'05*, ACM, San Diego, California.

Shah, R. & Rabaey, J. (2002). Energy Aware Routing for Low Energy Ad Hoc Sensor Networks, *Proc. of Wireless Communications and Networking Conference (WCNC)*, IEEE, Orlando, Florida.

Singh, S. & Raghavendra, C. S. (1998). PAMAS - Power Aware Multi-Access protocol with Signaling for ad hoc networks, *ACM Sigcomm Computer Communication Review* **28**(3): 5–26.

Sinha, A. & Chandrakasan, A. (2001). Operating System and Algorithmic Techniques for Energy Scalable Wireless Sensor Networks, *Proc. of International Conference on Mobile Data Management*, Hong Kong, pp. 199–209.

Sohraby, K., Minoli, D. & Znati, T. (2007). *Wireless Sensor Networks: Technology, Protocols, and Applications*, John Wiley and Sons, Inc.

Stevens-Navarro, E., Lin, Y. & Wong, V. (2008). An MDP-based Vertical Handoff Decision Algorithm for Heterogeneous Wireless Networks, *IEEE Trans. on Vehicular Technology* **57**(2): 1243–1254.

Stojmenović, I. (2005). *Handbook of Sensor Networks: Algorithms and Architectures*, John Wiley and Sons, Inc.

StrongARM (2010). Intel StrongARM SA-1110 Microprocessor.
URL: *http://bwrc.eecs.berkeley.edu/research/pico_radio/test_bed/hardware/ documentation/arm/sa*1110*briefdatasheet.pdf*

TinyOS (2010). .
URL: *http://www.tinyos.net/*

Van Dam, T. & Langendoen, K. (2003). An Adaptive Energy-Efficient MAC Protocol for Wireless Sensor Networks, *Proc. of International Conference on Embedded Networked Sensor Systems (SenSys)'03*, ACM, Los Angeles, California.

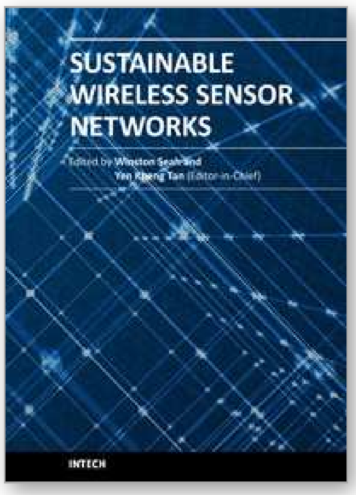
Varga, A. (2001). The OMNeT++ discrete event simulation system, *Proc. of European Simulation Multiconference (ESM)'01*, Prague, Czech Republic.

Ye, F., Zhong, G., Lu, S. & Zhang, L. (2005). GRAdient Broadcast: A Robust Data Delivery Protocol for Large Scale Sensor Networks, *ACM Wireless Networks (WINET)* **11**(2).

Ye, W., Heidemann, J. & Estrin, D. (2002). An Energy-Efficient MAC protocol for Wireless Sensor Networks, *Proc. of INFOCOM'02*, IEEE, New York, New York.

Ye, W., Heidemann, J. & Estrin, D. (2004). Medium Access Control with Coordinated Adaptive Sleeping for Wireless Sensor Networks, *IEEE/ACM Trans. on Networking* **12**(3): 493–506.

Yuan, L. & Qu, G. (2002). Design Space Exploration for Energy-Efficient Secure Sensor Network, *Proc. of IEEE ASAP*, San Jose, California.



Sustainable Wireless Sensor Networks
Edited by Yen Kheng Tan

ISBN 978-953-307-297-5
Hard cover, 574 pages
Publisher InTech
Published online 14, December, 2010
Published in print edition December, 2010

Wireless Sensor Networks came into prominence around the start of this millennium motivated by the omnipresent scenario of small-sized sensors with limited power deployed in large numbers over an area to monitor different phenomenon. The sole motivation of a large portion of research efforts has been to maximize the lifetime of the network, where network lifetime is typically measured from the instant of deployment to the point when one of the nodes has expended its limited power source and becomes in-operational â€“ commonly referred as first node failure. Over the years, research has increasingly adopted ideas from wireless communications as well as embedded systems development in order to move this technology closer to realistic deployment scenarios. In such a rich research area as wireless sensor networks, it is difficult if not impossible to provide a comprehensive coverage of all relevant aspects. In this book, we hope to give the reader with a snapshot of some aspects of wireless sensor networks research that provides both a high level overview as well as detailed discussion on specific areas.

How to reference
In order to correctly reference this scholarly work, feel free to copy and paste the following:

Arslan Munir and Ann Gordon-ross (2010). Optimization Approaches in Wireless Sensor Networks, Sustainable Wireless Sensor Networks, Yen Kheng Tan (Ed.), ISBN: 978-953-307-297-5, InTech, Available from: http://www.intechopen.com/books/sustainable-wireless-sensor-networks/optimization-approaches-in-wireless-sensor-networks