A CENTRALIZED ENERGY MANAGEMENT SYSTEM FOR WIRELESS SENSOR NETWORKS

by

Richard William Skowyra

A Thesis

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the

Degree of Master of Science

in

Computer Science

May 2009

Advisor: ______________________
Department Head: ______________

Abstract

This document presents the Centralized Energy Management System (CEMS), a dynamic fault-tolerant reclustering protocol for wireless sensor networks. CEMS reconfigures a homogeneous network both periodically and in response to critical events (e.g. cluster head death). A global TDMA schedule prevents costly retransmissions due to collision, and a genetic algorithm running on the base station computes cluster assignments in concert with a head selection algorithm.

CEMS' performance is compared to the LEACH-C protocol in both normal and failure-prone conditions, with an emphasis on each protocol's ability to recover from unexpected loss of cluster heads.

Table of Contents

Table of Figures

1 Introduction

Wireless sensor networks (WSNs) are increasingly deployed in a variety of environments and applications, ranging from the monitoring of medical conditions inside the human body to the reporting of mechanical stresses in buildings and bridges. In these and many other WSN applications the sensors cannot be recharged once placed, making energy expenditure the primary limiting factor in overall network lifetime.

One standard WSN configuration consists of a set of sensors that communicate to the external world via a base station, or sink, that has no power constraints. The sensors number in the hundreds or even thousands, and are primarily constrained by a limited battery supply of available energy. While the sink is modeled as a single node, it may provide access to other systems upstream such as distributed processing facilities or databases devoted to consolidating and cataloging the reported WSN data.

Since the primary form of energy dissipation for wireless sensors is in radio transmission and reception [1], a variety of network modifications have been proposed to limit radio use as much as possible. Some of these approaches focus on the MAC layer to minimize transmission costs (see Section 3.4.2.1), while other techniques operate on the network layer and attempt to minimize the range and duration of transmissions (see Section 3.4.2.2). Sensor clustering at the network layer has been shown to be a scalable method of reducing energy dissipation. Rather than individual sensor nodes transmitting their data to the base station, they instead transmit to another sensor designated as the local cluster head. The cluster head then sends aggregated (and possibly compressed) sensor information to the sink as a single transmission. Note that clustering makes some nodes more important than others, while increasing the energy dissipation of those same nodes.

This thesis implements a novel reclustering technique that minimizes both energy expenditure and loss of network coverage due to the failure of cluster heads. CEMS (Centralized Energy Management System) moves almost all processing not directly related to data collection off of the energy-limited sensor nodes and onto the sink. Furthermore, the base station maintains a record of expected transmission times from the network's cluster heads, based on their location on the global TDMA[1] schedule. If a cluster head consistently fails to transmit during its expected window of time, the sink triggers an emergency reclustering to restore network coverage. Two assumptions governed the creation of this system:

- The optimal clustering configuration changes over time as the residual energy of cluster heads decreases
- Any node, including a cluster head, has a non-zero probability of failing at a given time due to random accidents.

The sink maintains state information on each node in the network consisting of its location and its projected amount of residual energy. The assignment of nodes to cluster heads is calculated using a genetic algorithm (GA) which includes these factors when calculating the fitness of a legal potential clustering solution. The number of cluster heads, determined a priori, is an input parameter to the system.

CEMS was compared to the LEACH-C (See Section 5.1.2) protocol in a network configuration of one hundred nodes scattered over a 25m square field. While LEACH-C networks last slightly longer than CEMS networks under normal conditions, the latter outperforms the former when sensor failure rate is high due to random accidents. This can be ameliorated by using k-means clustering algorithm instead of a genetic algorithm, which was found to extend network lifetime by approximately 8%. Furthermore, CEMS restores coverage after cluster head death significantly faster than LEACH-C.

Before these experimental results are presented, related work is presented in Section 2. Background on wireless sensor networks, the simulation environments available for WSN research, and the role of energy dissipation in their design is presented in Section 3. CEMS itself is explained in detail in Section 4, and our experimental setup is presented in Section 5. Results are given and analyzed in Section 6.Conclusions and future work are detailed in Sections 7 and 8, respectively.

[1] TDMA, or Time Division Multiple Access, divides a period of time into slots. Each member of a TDMA network is assigned one time slot, during which it has sole access to the medium. Once the final slot's time interval expires, the schedule recycles to the first slot and the order repeats.

2 Related Work

Many protocols and algorithms to facilitate clustering in wireless sensor networks have been proposed in the past. LEACH, a self-organizing and adaptive protocol, is described by Heinzelman et al in [1]. Their work is directed at minimizing energy dissipation at the sensor level. LEACH takes a more distributed approach than CEMS and assumes that each sensor participates in an election process with its neighbors to determine a cluster head. This equates to a random distribution of cluster heads that, over time, equally distributes energy dissipation throughout the cluster. Determination of the optimal number of cluster heads is done a priori. A more in-depth description of this protocol is presented in Section 3.4.2.2.1.

A centralized version of LEACH, dubbed LEACH-C, is presented in [2] and explained below in Section 5.1.2. While the steady-state phases of both protocols are identical, LEACH-C uses a simulated annealing algorithm[2] running on the base station to calculate cluster assignments. This replaces the probabilistic self-election scheme used in LEACH.

Voigt et al.[3] extend LEACH-C to include solar-aware functionality. Sensors which are currently being charged via solar panels are more likely to be assigned the role of cluster head, and current cluster heads which cease to be charged may hand off their responsibilities to a new head mid-round.

Manjeshwar and Agrawal take a different approach than LEACH and its successors in [4]. Their TEEN protocol regulates both cluster assignments (using a round-robin cluster head selection scheme) and sensor reporting thresholds via user-configurable parameters to simultaneously minimize transmission cost and the overall number of transmissions. Super clusters (clusters of cluster heads) are used to extend the network's area beyond that of an individual sensor's transmission limit,.

A number of recent papers have been published which investigate the efficacy of genetic algorithms as tools to compute cluster assignments. In [5], Tang et al. propose using a genetic algorithm to determine a priori the optimal cycling of cluster heads in an implanted biosensor network. Their intent is not to ensure optimal energy usage, but instead to minimize heat dissipation into surrounding tissues over time. Thus, their fitness function includes a leadership history for each node, as well as the projected temperature increase that would result from its status as cluster head. The GA is only intended to be run once, with the resulting sequence repeated indefinitely.

Taking a more generalized approach, Mudundi and Ali [6] propose a genetic algorithm designed to find well-balanced clustering solutions for a wireless sensor network. Their primary fitness parameters are distance from the base station to the sensors in a cluster, the distance from cluster nodes to the cluster head, and transmission costs, and the number of cluster heads selected. The algorithm is reported to converge in 8 to 10 iterations, given 100 nodes.

Hussain and Matin [7] employ a genetic algorithm in one version of their Hierarchical Cluster-based Routing (HCR) protocol. Fitness is determined by node density, distances between sensors, cluster heads and the sink, transmission costs, and the number of expected transmissions that round, among other factors.

Despite the amount of work put into protocols designed to minimize WSN energy expenditures, little research has been conducted on the robustness of these systems in real-world conditions. While a simulation environment excels at establishing baseline performance metrics, sensors are often subject to a variety of field conditions which are not modeled during simulation. Specifically, a number of accidents, from deliberate vandalism to storm damage, may disable or remove sensors from the field. This affects not only the coverage of individual sensors, but may eliminate the routing path of an entire cluster if a cluster head is rendered inoperative. The ability to rapidly reintegrate sensors which have been removed from a network due to cluster head failure is not a design point in any of the above protocols, but could prove valuable in real-world conditions.

3 Background

Wireless sensor networks have been an active topic of military research and development since the 1980s, with a number of DARPA projects funding their development. The Distributed Sensor Network project [12], initiated in 1980 and spanning a range of topics from signal processing to distributed computing and tracking

[2] Simulated annealing is a Monte Carlo method that probabilistically replaces the current solution with a solution nearby in the search space, based on the difference between their scores and a global constraint which is gradually tightened each iteration.

technologies, was the first consolidated effort to develop a wireless sensor platform. Due to the state of technological miniaturization during the 1980s, however, these early sensor networks generally used large, powered sensors deployed in a hierarchical network to share data among a number of sites. The U.S. Navy's Cooperative Engagement Capability [13], for example, combines the sensor measurements of a number of independent ships to form a coherent description of airspace over a given region. This is accomplished by creating a shared database of sensor data from a geographically distributed network of mobile, ship-based sensors.

The modern incarnation of WSNs as distributed networks of small, low-cost and low-power sensors did not arise until the early 21st century, when technological advances in MEMS (Micro-Electromechanical Systems) allowed for cheap mass production of wireless sensors. This spurred an ongoing research effort in the academic and commercial communities to both improve the capabilities of WSNs and extend their applications to an ever-widening array of situations. Today wireless sensor networks exist in industries disparate as healthcare, civil engineering, agriculture, industrial automation, traffic control, and security surveillance.

3.1 Characteristics of a Wireless Sensor Network

While WSNs have much in common with more traditional ad-hoc and infrastructure-mode wireless networks, they differ in several important ways. A WSN generally has a large number of sensors scattered over an area and a single node referred to as the base station, or sink, which is responsible for receiving data transmitted by sensors in the field. It bears some similarity to an access point in an infrastructure-mode network. The sink may or may not be located inside of the space being sensed, and is almost always considered to be a powered node operating without energy constraints. Depending on the application and configuration of the WSN, the base station may have additional responsibilities such as coordinating network activities, processing or formatting incoming data, or working with an upstream data analysis system to provide data matching any query requests that it receives.

In contrast to the single base station, WSNs may have hundreds or even thousands of sensors operating in the field. These low-power devices are often battery powered and sometimes include solar panels or other alternative energy sources. During their limited lifespan (defined as the time interval during which sufficient energy remains to transmit data), sensors are tasked with monitoring a single aspect of their surrounding environment and reporting their sensed data via an onboard radio transceiver.

Given the above characteristics, a few important differences from standard WLANs become apparent:

- Most traffic is upstream, from the sensors to the base station.
- The small amount of downstream traffic tends to be dominated by broadcast traffic from the base station to provide generic updates to all sensor nodes.
- Power use is a key performance metric, as sensors are battery powered.
- Network links tend to be low-capacity, as high throughput is energy intensive and often unnecessary.
- Long delays may be acceptable for many WSN applications (e.g. a network monitoring soil pH will be relatively immune to high latencies.)

3.2 Components of a Wireless Sensor Network

WSNs employ a variety of hardware platforms and software systems. Sensors themselves vary in size from a few millimeters (e.g. Dust Network's DN2510 [14]) to the size of a PDA (Crossbow Technology's Imote 2 [15]) or larger (Harvard University's CitySense sensors [16]). Even within similar sensors, radio transceivers, sensors, and microprocessor facilities may vary. Given the wide variety of platforms available, any protocols developed for a WSN must consider the characteristics of the underlying hardware on which they will operate.

3.2.1 Hardware

Wireless sensor nodes have shrunk significantly as MEMS technology has progressed. Individual components are now often integrated into the same chip, and hardware design has evolved to reflect this change. A sensor node is composed of several independent components linked together to form one operative package. A power supply provides the necessary energy to a sensing unit designed to monitor the environment and produce a representative signal, a processor governing sensor operations, onboard flash memory, and the radio transceiver responsible for linking the sensor node to the rest of the network.

3.2.1.1 Sensing Unit

At their simplest, sensors are designed to generate a signal corresponding to some changing quantity in the surrounding environment. This may be anything from a simple Peltier diode to measure temperature (such as the Microchip TC74 [17]), or as complex as a charge-coupled device to monitor video input (such as the Omnivision OV7640 [18]). The data are sent to an onboard microprocessor after being converted to a digital signal by the sensor electronics, which may perform some processing or culling before electing to transmit the information over the module's transceiver. Generally sensors are procured and attached to WSN sensor platforms by the purchaser, and are often manufactured by different companies than those which provide the platform itself.

3.2.1.2 Processing Unit

A sensor's processor must, at minimum, serve as an effective interface to the sensor module and regulate data flow from the sensing unit to the radio transceiver. There are currently three popular types of processing unit in general use: microcontrollers, microprocessors, and Field-Programmable Gate Arrays (FPGAs). Onboard storage, often in the form of flash memory, is also often included as part of a sensor's processing unit.

Microcontrollers such as the 8-bit TI MSP430 [19] and 16-bit Atmel AVR [20] are the simplest and one of the most common forms of processor, unable to support complex operations but running at a low clock speed and consuming the least amount of power. They are most often used when little data processing or decision making is necessary.

Microprocessors such as the 32-bit Intel Xscale [21] are a more general-purpose CPU, and are potentially much more powerful than microcontrollers, with significantly higher clock speeds and more flexibility in terms of their programming. This does come at a commensurately increased energy cost, however.

Field-Programmable Gate Arrays (FPGAs) use a hardware description language to allow sensor modules to be reconfigured in the field to rapidly process the data that their sensor units are reporting. This can be invaluable for real-time surveillance networks and target tracking, where image processing algorithms can be implemented on the hardware level without purchasing a dedicated GPU. FPGAs are also the highest energy consumers of the three processors, and may not be compatible with general-purpose WSN software systems.

3.2.1.3 Radio Unit

Wireless sensor networks operate primarily in the 430MHz, 900MHz, and 2.4GHz ISM bands. Individual transceivers are tunable within their designated band, and several (such as the CC1021 [22]) can toggle between multiple frequency bands. Depending on the transceiver, maximum bit rates as low as 76.8bps (e.g. the CC1000 [23]) or as high as 250bps may be supported.

Regardless of frequency and bit rate, radio transceivers are the primary energy consumer in any wireless sensor node. For this reason the vast majority of modern transceivers (such as the Chipcon CC2500 [24]) provide onboard hardware support for several discrete states of operation: transmit, receive, idle, and sleep. Transmitting consumes the most power, but reception of a broadcast, even one not intended for the node in question, has a nontrivial power cost (see Section 3.4.1). Idle modes turn off the radio oscillator, but generally use a polling or carrier sense mechanism to check the medium periodically. This dissipates less energy than leaving the transceiver in receive mode, but is sufficiently expensive that many protocols have evolved which avoid the idle state. A radio transceiver operating in sleep mode is essentially turned off, consuming negligible amounts of power but unable to interact with the network at all.

Despite these operative modes being referred to as discrete states, real world electronics are obviously unable to switch modes instantaneously. This delay is one of the major attractions of using a sensor's idle mode over its sleep mode, as wake up times are significantly less in the case of the former. Switching between transmit and receive states is generally quite fast (on the order of a few nanoseconds), but switching to and from idle and sleep states generally requires times in the hundreds of nanoseconds. In the case of sleep mode , power-on and calibration delays must also be accounted for in energy consumption calculations.

3.2.2 Software

Software written for wireless sensor networks differs from more conventional platforms in several respects. The vast majority of WSN operating systems and network protocols that have been produced in academia or in the commercial sector are power-aware due to the limited amount of energy available to sensor nodes. From a software design standpoint, this necessitates optimizing algorithms and program architectures to minimize the amount of energy dissipated per operation. Efficiency of execution in terms of running time is still a concern, but is of secondary importance. If an algorithm could finish rapidly but consume more power than a slower implementation, the slower version might still be selected for use in a WSN.

Furthermore, sensor nodes are extremely limited in terms of resources. On-board RAM capacity is extremely small due to the energy drain of volatile memory. Software systems must therefore rely primarily on register-based operations and any flash-based storage medium that might be present. This requires that a program use a limited number of often-accessed data structures, and that it performs computations using as little memory as possible.

Two other distinguishing features of WSN software arise less from hardware limitations and more from environmental constraints. Since sensors can be deployed in a potentially inaccessible field (e.g. underwater, inside walls, in a combat zone), WSN software systems must be able to run unattended for long periods of time. Any logical or physical faults should be able to be dealt with, worked around, or minimized in impact without the intervention of human agencies. Support for any kind of graphical user interface, or even a terminal interface in field conditions, is not generally provided. Sensors may be reprogrammed or configured in controlled conditions, however, via software running on an external machine to which individual nodes may be connected.

3.2.2.1 Operating Systems

WSN-specific operating systems are distinguished from existing embedded OSs such as ChibiOS/RT or Nucleus RTOS by their lack of real-time processing constraints. Sensor networks are rarely interactive, and only a few specific applications such as live video surveillance impose any strict time constraints on data acquisition and processing. Since sensor hardware is often extremely limited in terms of both resources and available energy, small footprints and efficient use of memory and processor cycles is a key requirement for any WSN operating system. An example of how these requirements guide WSN operating system design can be seen in TinyOS.

3.2.2.1.1 TinyOS

Originally developed by the University of California, Berkeley, and Intel Corporation in 2002, TinyOS [25] has become one of the most popular WSN operating systems available. Today it is maintained by an international community of developers and users, the TinyOS Alliance.

TinyOS is implemented in nesC (network embedded systems C), a dialect of C developed in the early phases of TinyOS's design and made specifically to optimize programs for use in a wireless sensor network. It provides static, compile-time race detection and a number compiler optimizations designed to dramatically reduce an application's footprint. In order to provide these optimizations, nesC does not allow the declaration of function pointers or the use of dynamically allocated memory.

The developers of TinyOS used two principles to direct its development:

- The system is event-centric rather than process-centric. Computation occurs reactively through event handling, similar to the mechanisms employed in many network simulators. The reasoning behind this principle is that sensor networks are themselves event-based: data is measured either periodically or reactively, processed, and sent to the base station.
- The system is a platform for innovation. Rather than making TinyOS the optimal choice for a specific application or subset of applications, the operating system is designed to be flexible and easily configured for a given role.

The latter principle guided the operating system's overall architecture, while the former influenced the development of the component model.

TinyOS is as an architectural framework and collection of components which can be used to easily make an application-specific operating system. These components include not only standard hardware interfaces and services (e.g. sensor components, system clocks, timers, etc.) but also a variety of advanced power management and network control components. These are designed to minimize energy use and optimize duty cycles for a variety of common scenarios.

Each component is an independent module which exposes a set of split-phase interfaces, which decouple requests for service from the results of those requests. A configuration file referred to as a wiring specification links the interfaces of components to those of other components, defining what operations may be requested and where data may be sent. Components may be wired in such a way as to create a super-component, such as a network stack.

Individual components in TinyOS correspond to a specific set of services, and often map to individual hardware modules such as a system clock or sensing unit. These services are accessed asynchronously via commands, events, and tasks. TinyOS defines a command as a function that is implemented by the component which provides a given interface, such as a sensor module with a getData command. A component which finishes executing a command triggers an event containing its results (e.g. a dataReady event containing the sensed data). Events are implemented by the users of a given interface, such as a processing module that performs compression or aggregation of sensor data. Finally, tasks represent deferred computation within a single component. They may be placed in a component's scheduler by either commands or events, and are executed in a FIFO order. Each task is run to completion before the next is executed, so tasks may be considered atomic with respect to one another.

By wiring the correct components together, TinyOS users can create an operating system specifically optimized for their sensor node hardware and their application. nesC attempts to ensure that the resulting code footprint will be as small as possible: the TinyOS kernel is 400 bytes, with most applications ranging from less than 16KB to approximately 64KB.

3.3 WSN Simulation

Wireless sensor network research is largely directed at improving the energy efficiency, coverage, reliability, and security of sensors and networks. This translates into a need for detailed information about conditions on the lower levels of the network stack in an ad-hoc wireless environment. Conventional network simulators often have more support for packet-level network-layer simulation than for, e.g., frame-level information gathering and radio energy dissipation modules. To meet this need a number of simulators have evolved or adapted to service the needs of WSN research.

3.3.1 GloMoSim

GloMoSim [26] is the academic version of the commercial Qualnet wireless network simulator. It is written in Parsec, a dialect of C designed for discrete event simulation.

GloMoSim only currently supports simulation of wireless networks, with no modules included for sensor-oriented research. Networks can easily scale up to thousands of nodes, however. The simulator's network stack is also designed to be easily configurable, with modules on each layer easily swapped out if the interfaces are preserved. Each layer has built-in statistics collection.

3.3.2 ns-2

The de facto simulator for wired networks, ns-2 [27] now includes support for a wide range of wireless protocols. Modules, generally defined as individual protocols, to be used in the simulation are written in C++, and user command scripts are written in OTcl. These scripts define network topology, relationships between modules, and how simulation data is output. The Zigbee 802.15.4 standard is supported, and advanced radio propagation modules are available. In addition, there is a significant amount of support available in the form of mailing lists, online manuals and tutorials, and message boards.

3.3.3 Omnet++

Omnet++ [28] is a discrete event simulator designed for wireless mobile and ad-hoc networks. Like ns-2, Omnet++ uses modules written in C++. Rather than OTcl, however, it uses the NED topology description language to define topologies and relationships between modules. On its own Omnet++ has only limited support for WSN research. However, the MiXiM framework, a merger of the Mobility Framework, MAC Simulator, ChSim, and Postif frameworks, provides extensive support for physical, MAC, and network-layer simulation in which frame level information can be easily collected and analyzed. Many existing WSN MAC protocols, such as S-MAC, SC-MAC, and L-MAC are included. The Omnet++ simulator is much more extensively used in the European academic community than the American community. This can make finding support difficult if the existing tutorials are not sufficient. A mailing list is available, but response times tend to be slow.

3.3.4 TOSSIM

TinyOS includes TOSSIM [29], a sensor node simulator that also uses nesC. TOSSIM is not a general-use WSN simulator, but is instead intended to allow for design and troubleshooting of TinyOS networks. All state information of all nodes, in addition to network information, is available to the user. Since TOSSIM simulates nodes running TinyOS, a simulation can actually interface with applications running on a base station as though it were a physical network. This is the standard simulator in use for networks based on TinyOS applications, but has little practical use outside of the TinyOS community.

3.4 Power as a Limiting Factor

By their nature, wireless sensor networks are limited in their operative lifespan by the amount of energy available to sensor nodes. Recent advances in MEMS and NEMS (micro- and nano- electromechanical systems) technologies may allow for greatly increased battery energy densities in the coming years, but without a reliable method of replenishing them even the most powerful energy supplies will dissipate and a sensor node will eventually go offline.

A popular trend in both simulated and real-world sensor networks is to mount energy-harvesting components on some or all of a network's sensor nodes, allowing their power supplies to be replenished in the field. Depending on the environment where sensors are deployed, these energy harvesting mechanisms may use a variety of techniques:

- Solar harvesters (e.g. the Heliomote [30]) employ solar cells to charge a battery or supercapacitor when sunlight is available. Due to the relatively low efficiency (in terms of power output to surface area) of solar cells, however, solar energy harvesting does not provide reliable power for small sensor nodes.
- Wind generators (e.g. Ambimax [31]) use small turbines moved by air currents to generate energy. While impractical in many settings, wind power can be useful for sensors mounted on bridges, building exteriors, etc.
- Mechanical harvesters convert mechanical stress (either strain or vibration) to electrical energy using, e.g., piezoelectric membranes. This can be a useful source of power for sensors on bridges, railroad and subway tracks, and any other environment which is subject to movement.
- RF harvesters use recent advances in wireless power transmission to recharge sensor nodes using broadcast power from a nearby transmitter. While limited in range, these systems have the potential to remove the energy constraints traditionally applied to WSNs.

Hybrid approaches are also occasionally employed when cost and size constraints allow. Solar and wind harvesters are especially complementary, as windy conditions tend to coincide with cloudy conditions.

Topological techniques are also sometimes employed to conserve node energy. Rather than deploying a homogenous network of sensors, certain designated gateway nodes may be included. These are generally attached to energy harvesters, and are responsible for routing long-range communications between the sensor nodes and the base station. Other networks use a combination of static, battery-powered sensors and mobile sensors equipped with energy harvesters. These mobile nodes are able to move throughout the field and recharge static nodes, distributing energy gains amongst all members of the network.

3.4.1 Radio Energy Dissipation

The primary energy consumer in any sensor node is its radio transceiver. Precisely how much energy is dissipated per bit transmitted or received is a function of distance, bit rate and transmission length, among other factors. A significant amount of research has been conducted with the intent to minimize the duration and distance that signals must be transmitted, as well as the number of extraneous signals that must be received. In order to quantify the performance of these power-aware protocols, the behavior of radio wave propagation must be accurately modeled. This is a limiting factor in many network simulators, as such physical layer concerns have traditionally been in the purview of electrical engineers and not a factor in protocol design. Several models of increasing complexity exist, each of which is useful in certain scenarios.

3.4.1.1 Free Space Model

The free space model of radio wave propagation is by far the simplest. In the free space environment, signal strength (in watts or decibels) falls as a power of distance traveled from the transmitter. The simplest application of this is the well known inverse-square law, which states that a signal's power is inversely proportional to the square of the distance from the source. When applied to a wireless medium, the free space

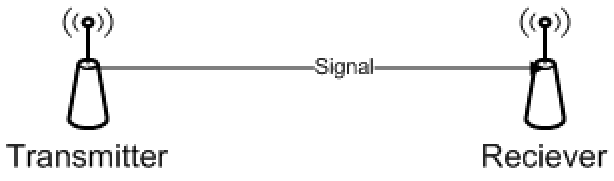

Figure 1 - Free Space Model

model can be described in terms of the relationship between transmitted power, P_t, and received power, P_r [32]:

$$\frac{P_r}{P_t} = G_t G_r \left(\frac{\lambda}{4\pi d}\right)^2$$

- G_t – Transmitter antenna gain
- G_r – Receiver antenna gain
- λ – Signal wavelength
- d – Transmitted distance

This model assumes that path loss is the only force acting on the radio signal. Reflection (attenuation and redirection of a signal), diffraction (strong attenuation and signal splitting in a cylindrical space), and scattering (attenuation and propagation in all directions) are ignored. The signal therefore takes a single path from the transmitter to the receiver; experiencing no interference and encountering no obstacles (see Figure 1). While useful for approximations at small distances, the free space model fails to model real-world conditions accurately enough to be relied upon in a simulation of power-aware systems.

3.4.1.2 Two-Ray Ground Reflection Model

The two-ray ground reflection model is more complex than the free space model, but more accurately models real-world settings. While still extremely simplistic, it assumes that both transmitter and receiver are elevated some distance above the ground. In addition to the direct path from transmitter to receiver, a reflected signal off the ground constitutes a second signal path that either constructively or destructively interferes with the original signal at the receiver (see Figure 2.) [32] :

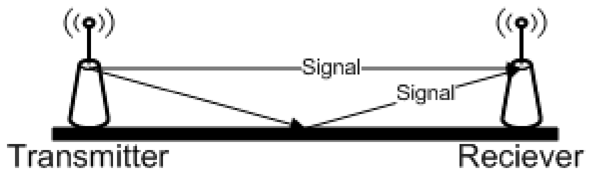

Figure 2 - Two-Ray Ground Reflection

$$P_r = P_t G_t G_r \left(\frac{h_t^2 h_r^2}{d^4}\right)$$

- G_t – Transmitter antenna gain
- G_r – Receiver antenna gain
- d – Transmitted distance

- h_t – Transmitter height
- h_r – Receiver height

Note that in addition to height being accounted for, the path-loss exponent has increased from two to four. This significantly impacts the range of low-power transmitters such as wireless sensor nodes, and is a much more realistic value for representing real-world conditions. Two-ray ground reflection still fails to account for any obstacles other than the earth itself. However, in most environments there will be trees, walls, urban features, mobile obstructions, and other factors which cause signal reflection, scattering, or diffraction.

3.4.1.3 Shadow Fading Model

The path-loss exponents used above do not take the density and material of obstacles into account. In real-world settings, a variety of constructed and natural materials will interfere with a signal enroute from the transmitter to the receiver. Mobile obstacles, such as human beings, vehicles, moving tree leaves, etc. may also be present. The effects of such sources on a signal are referred to as shadow fading, and result in the same signal being received at different strengths at the same distance from the transmitter. Over time the received signal strength tends to fluctuate around a given mean, as obstacles move in and out of range. Taking this fluctuation into account, path loss can be modeled by the following equation [32]:

$$L_p = L_0 + 10\alpha \log(d) + X$$

- L_p – Path loss in decibels
- L_0 – Path loss at 1 meter
- α – Path loss exponent. For most 2.4GHz applications, $\alpha = 4$
- d – Distance from transmitter to receiver
- X – A random variable with a distribution appropriate to the environment. This can be determined empirically by regression analysis of received signal strengths.

Note that this model also fails to explicitly model the effects of radio wave diffraction, scattering, and reflection off of multiple obstacles. While it is accounted for to some degree in the random variable X, this is a purely statistical approach to the problem.

3.4.1.4 Multipath Fading

Advanced radio propagation models also account for the effects of multipath fading. This effect arises from the same signal intersecting with multiple obstacles and traveling different distances from transmitter to receiver as it is reflected, diffracted, and scattered off of various surfaces. The individual signals therefore not only arrive at slightly different times and levels of attenuation, but also in different phases. This causes rapid fluctuations of received signal amplitudes, and leads to extremely high bit error rates. A number of mechanisms, such as directional antennas, coding, and channel or spatial diversity, are used to mitigate the effects of multipath fading. Only a few simulators, such as Omnet++ [10] (with the Mobility Framework or MiXiM) and ns-2 provide sufficiently detailed radio propagation models to account for multipath fading.

3.4.2 Energy Minimization

A significant amount of research in WSNs has been on power-aware protocols and energy minimization strategies. These can broadly be grouped into two classes: those which attempt to minimize the overhead and power cost of network communications on the MAC layer, and those which attempt to minimize the distance and duration of communications via network-layer routing protocols. The most common form of protocol in the latter category is clustered routing, though other techniques such as beaconing are also employed.

3.4.2.1 MAC-Layer Protocols

There are a significant number of MAC protocols for wireless sensor networks available from both academic and commercial sources. They range in scope from TDMA-based protocols to CSMA and combinations of the two. In general, the WSN MAC protocols attempt to minimize energy dissipation through one or both of the following strategies:

- Collision avoidance – Not only do retransmissions dissipate potentially large amounts of energy, but detection of the collision may itself be expensive.
- Overhearing avoidance – Reception of packets, even those not intended for the node in question, dissipates energy operating the radio’s receiver electronics and processing the signal. This is especially draining on sensor power supplies

in dense networks, where one broadcast may reach many sensors and many broadcasts may therefore be received in a short amount of time.

A number of other problems exist in specific types of WSNs, such as dense deployments or delay-sensitive applications. Crankshaft, a relatively new protocol that is still under active research, attempts to mitigate the challenges associated with overhearing in the former case.

3.4.2.1.1 Crankshaft

Crankshaft [33] is a relatively new MAC protocol, and represents something of a synthesis of the SCP-MAC and L-MAC protocols. It uses a combination of TDMA-based frames and slots and CSMA-based contention resolution to ensure that a minimal amount of latency is introduced while increasing energy efficiency by minimizing overhearing and retransmissions. The protocol's authors designed it specifically for dense sensor networks, where overhearing is the primary energy drain on sensors.

Time in Crankshaft is divided into frames, and each frame is divided into unicast slots followed by broadcast slots. During any given unicast slot a specific node is listening for incoming transmissions. All nodes know the slot that other nodes listen on, as each slot offset is a modulo function of a node's MAC address and the total number of slots. If a node wishes to transmit to another node, it wakes up during the receiver's slot and checks the medium for contention. If the channel is clear the message is transmitted; otherwise the sender backs off and has a 70% probability of retransmitting each frame for three frames. During broadcast slots all nodes listen to the medium, and any message that is transmitted is assumed to be directed to all members of the network. The sink listens during all slots in the frame, as the vast majority of messages are assumed to be directed at it and it is not likely to be energy-constrained.

Crankshaft performs well against existing WSN MAC protocols in terms of energy efficiency at loads of up to 80%, and in terms of delivery ratio in loads of up to 50% (after which it is surpassed by LPL). A significant amount of latency is introduced, however, making it potentially unsuitable for delay-sensitive applications such as object tracking.

3.4.2.2 Clustering Protocols

While clustering protocols may also operate on the MAC-layer, many can be implemented over an existing WSN MAC implementation. These protocols attempt to minimize the distance a sensor must transmit as well as the duration of each transmission. The former objective is accomplished by dividing a field into a number of cluster domains, each of which is administered by a cluster head (which may be a normal sensor node, or may have special features such as an energy harvester). Members of each cluster send data to their local cluster head using a mechanism defined by either the underlying MAC layer or the clustering protocol itself. Once all members of the cluster have reported in, the cluster head generally performs an application-specific form of aggregation and/or compression on the messages before forwarding them to the base station. For large networks, super-clusters composed of cluster heads may be used to introduce a form of hierarchical routing. Similarly, cluster heads may be used to form a backbone network of high-capacity, long-distance links.

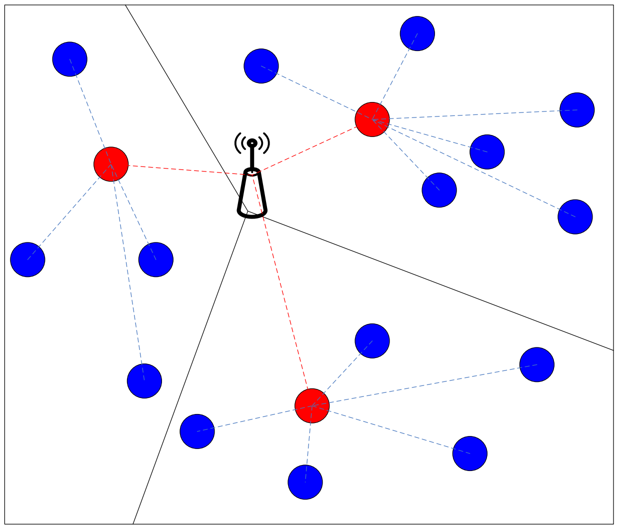
Figure 3 - WSN Clustering

Note that clustering makes some nodes more important to maintaining routing paths than others, while simultaneously increasing the energy dissipation of those same nodes. For this reason, mechanisms to rotate cluster heads within a cluster (and potentially redefine members of each cluster) are key aspects of any clustering protocol. This rotation may be a distributed function which operates over each sensor node, or may be centralized at the base station. The Low Energy Adaptive Clustering Hierarchy (LEACH) is an early clustering protocol which employs the former technique. It has become the standard of comparison against which new protocols are often evaluated.

3.4.2.2.1 LEACH

Developed by Dr. Wendi Rabnir Heinzelman in 2002, LEACH [1] uses a periodic distributed clustering function to balance energy costs throughout the network. Time is divided into rounds, and every sensor has a certain chance of self-electing itself as a cluster head. The specific probability is a function of the optimal number of cluster heads (determined *a priori*), the number of nodes in the network, and the node's energy level relative to the aggregate residual energy in the network. This attempts to ensure that high-energy nodes are cluster heads more often than low-energy nodes.

Once a node has self-elected itself, it broadcasts a message to the surrounding sensors. Each sensor sets its cluster head to the node ID of the strongest signal that it received, and the network's data-collection phase begins. This process is repeated each round.

In order to minimize retransmissions and overhearing, LEACH uses intra-cluster TDMA. This necessitates dynamic TDMA cycles, however, which are difficult to implement on a hardware level. Each cluster uses a different CDMA spreading code to ensure that adjacent networks do not interfere with one another, and cluster heads contend for transmission privileges to the base station using CSMA.

4 The Centralized Energy Management System (CEMS)

The Centralized Energy Management System is a clustering protocol that exploits the predictable nature of TDMA-based channel access to rapidly detect and respond to critical failures. Almost all energy-intensive operations (such as

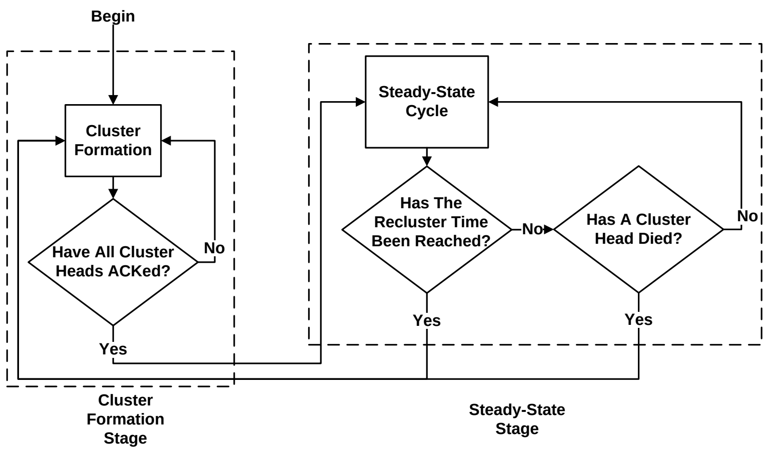

Figure 4- CEMS Overview

cluster formation) are moved upstream to the base station, which is assumed to not have any energy constraints. CEMS has two distinct phases: cluster formation and steady-state operation. The former is run at the beginning of each reclustering phase, which occurs both periodically and in response to cluster head death. The base station calculates cluster assignments and notifies the new cluster heads. If all heads acknowledge, the steady-state phase is initiated. During this phase, sensors report data to their cluster heads. The data are then compressed and aggregated before being forwarded to the base station.

Two assumptions governed the creation of this system:

- The optimal clustering configuration changes over time as the residual energy of cluster heads decreases
- Any node, including a cluster head, has a non-zero probability of failing at a given time due to random accidents.

The sink maintains state information on each node in the network consisting of its location and its projected amount of residual energy. The assignment of nodes to clusters is calculated using a genetic algorithm (GA) which considers nodes' spatial positions, and the assignment of a cluster head to each cluster is calculated using node energy and position. The number of cluster heads, determined a priori, is an input parameter to the system.

4.1 Clustering Phase

CEMS' clustering phase is initiated at network startup and at each subsequent reclustering, whether due to period triggers or in response the cluster head failure. Selection of cluster heads and cluster members is divided into two stages. A genetic algorithm first determines cluster membership for each sensor in the network during the cluster formation stage. This information, along with spatial coordinates and current energy levels, is then passed to a cluster head selection algorithm during the head selection stage. Once both cluster heads and members have been determined, the sink informs each sensor of its new assignment during the sensor notification stage.

4.1.1 Cluster Formation

The genetic algorithm which determines cluster membership is implemented with the GALib C++ library. It uses a fixed-length list of integers to describe a genome representing a potential network topology. The genome's length is always equal to the current number of living sensors. Each value in the list signifies a cluster ID. (The number of clusters is determined *a priori*.) The index of each cell represents a unique sensor in the network. Since the network's population of active sensors will change over time, sensor IDs are referenced indirectly through a lookup table. An example of this can be seen in Figure 5.

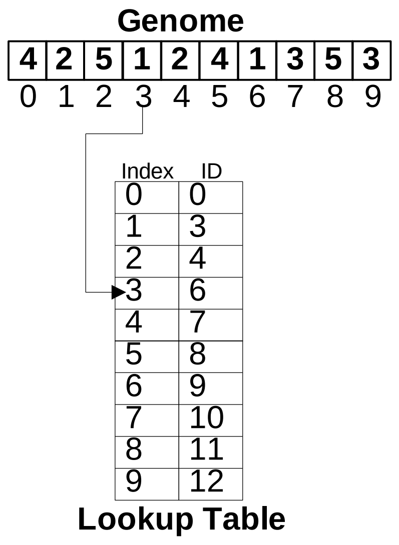

Figure 5 - Genome Representation

Selection is accomplished through the minimizing objective function presented in Figure 6. First, a centroid[3] for each cluster is determined. Each cluster is then assigned a score based on the sum of the squared distances between each cluster member and that cluster's centroid. The sum of all cluster scores is used as the objective score for that genome.

$$z_c = \frac{(z_{s_1} + z_{s_2} + \ldots + z_{s_k})}{k}$$

$$s_z = \sum_{i=0}^{n} \left(\sqrt{(z_c - z_{s_i})^2}\right)^2$$

$$Obj = \sum_{k=0}^{m} s_k$$

z_c – The centroid for cluster z
z_{si} – Sensor i in cluster z
s_z – Score for cluster z
n – The number of sensors in a given cluster
m – The number of clusters in the genome

Figure 6 - Objective Function

Each individual sensor has a probability of being chosen for mating equal to its fitness score divided by the sum of fitness scores over that generation. Two individuals are chosen each generation, and the highest scoring genome is selected.

[3] A centroid is the geometrical center of a set of points, determined by averaging all of the spatial coordinates of the points comprising the set.

4.1.1.1 Configuration

The original settings for single-point crossover and mutation values were based on the recommendations published in [8] and [9]. The parameters' final values were arrived at via tuning based on empirical tests in the Omnet++ discrete event simulator [10]. While the configuration used in our experiments did cause the algorithm to converge to a reasonable objective score, the process consumes a nontrivial amount of time. Note that the parameters we selected were chosen with the resources of a high-performance computer in mind. Practical hardware limitations may necessitate tuning the algorithm’s parameters.

Population size per generation was set to one hundred genomes. This is something of a tradeoff between a large set of representative genomes, which allows a more informed search of the space, and execution time. Larger populations obviously take longer to evaluate per iteration, but also reduce the probability of premature convergence to a local minimum.

Mutation rate was set to 0.0005. While this is a rather small probability, the fact that one hundred genomes with (initially) one hundred chromosomes will be present per generation puts this figure in some perspective. Lower mutation rates tend to fail in their primary task of acting as a secondary search parameter designed to prevent premature convergence to local minima. Higher mutation rates remove much of the evolutionary behavior of a genetic algorithm, and begin to cause the GA to perform more like a random search.

Crossover rate was set to 0.06. As the primary search parameter, crossover rates must be balanced between high values which produce and discard superior genomes before they can be evaluated, and low values that cause stagnation and limit the search space to a subset of the actual region.

Finally, the number of generations to iterate through before termination was set to 5000. This is a fairly high value, but empirical analysis of objective scores over time did not indicate significant convergence earlier. It is possible that tuning of the above parameters could result in a faster rate of convergence, and allow this number to be lowered. Doing so could significantly improve runtimes of the genetic algorithm.

4.1.1.2 Scoring Problems

During the early phases of the genetic algorithm's development, its objective function considered several factors beyond spatial location and attempted to determine both cluster membership and each cluster's head. In order to do so sensor energy levels, position relative to other sensors, position relative to the base station, cluster population, projected transmission costs, and projected reception costs were all considered when calculating the objective score of each genome. Unfortunately, this created a situation in which a low score could be achieved by sacrificing some scoring components in order to minimize others. Seemingly good solutions, for example, would be returned in which every sensor was its own cluster head, cluster heads were members of multiple clusters, or cluster heads were far from their members but adjacent to the base station. In order to curtail this behavior, head selection was divided into two stages midway through CEMS's development. In doing so, however, several benefits of using a genetic algorithm were lost. Since a single value is now being minimized, a K-means clustering algorithm might be both a faster and more efficient method of computing cluster assignments.

4.1.1.2.1 K-Means Algorithm

The k-means algorithm partitions *n* objects, in this case sensor nodes where each node is a 2-dimensional structure (representing x- and y-coordinates in the field), into *k* clusters. This is most frequently done by iterative refinement over a random (or heuristically determined) initial assignment. Each subsequent iteration potentially reassigns objects to different clusters, attempting to minimize the sum of the squared distances from each object to the centroid of its cluster. This is not dissimilar to the objective function described above, although k-means does not generally use an evolutionary technique during iterative refinement. The algorithm runs much faster than most GA configurations are able to, but is also not guaranteed to converge to a global minimum.

4.1.2 Head Selection

Once individual sensors have been assigned to clusters, a head selection algorithm elects one sensor per cluster to the position of cluster head. Each cluster is searched over for the sensor with the highest residual energy. All sensors whose remaining energy level is within 5% of that sensor's energy level are put in a privileged subset from which the cluster head will be drawn. The sensor which is both a member of that subset and closest to the centroid of the cluster is then chosen as a cluster head for that round.

Note that the 5% energy threshold was arrived at empirically. Depending on the actual topology of the cluster, another value may exhibit better overall performance. Clusters covering a large area, for example, might increase the threshold and thereby place more importance on spatial location at the expense of fair balancing of energy load. Conversely, clusters which cover a small area might emphasize fair balancing and reduce the threshold even further.

Outlying sensors can further complicate head selection, as the current algorithm will favor a fully charged sensor that is far from the centroid over a closer sensor with lower battery levels. This increases the transmission cost for all sensors which must transmit to the outlying cluster head, potentially wasting more energy than is saved by using a more fully charged sensor. This scenario only occurs, however, if the clustering algorithm is unable to efficiently cluster sensors due to unavoidable topological concerns or convergence at a local minimum.

4.1.3 Sensor Notification

After sensors have been assigned and heads elected to clusters, the base station broadcasts a message to each cluster head informing it of its new role, which sensors are in its cluster, the distance it must transmit to the base station, and the distances that its members must transmit. The cluster head relays distance and membership data to each sensor in its cluster and sends an acknowledgement to the base station. Once all acknowledgments are received, the sink initiates the network's steady-state phase.

If all cluster heads do not send an acknowledgement before a timeout window expires, the sink reclusters and increases the missed transmission count of any cluster head which failed to acknowledge. Any sensor with three consecutive missed transmissions will be declared dead and removed from future clustering assignments.

4.2 Steady-State Phase

The steady-state phase uses a layer-3 protocol which employs cross-layering to control sensor radio states. All nodes in the network share a single TDMA cycle, with a number of slots equal to the initial population of the network. The number of slots in this cycle will never decrease, despite sensor deaths creating unused slots as time goes on. The reason for this design is explained in Section 4.2.1.1

During this phase, sensors periodically report data to their cluster head. Depending on the data type, the head aggregates and/or compresses its members' messages before relaying them to the base station. This process continues until the reclustering period expires and the sink reclusters to balance energy loads, or until a cluster head dies and the sink initiates an emergency reclustering. An overview of this process is given in Figure 7. Note that the steady-state phase of the network is significantly longer than the clustering phase; almost all of a network's lifetime will be spent in this mode of operation.

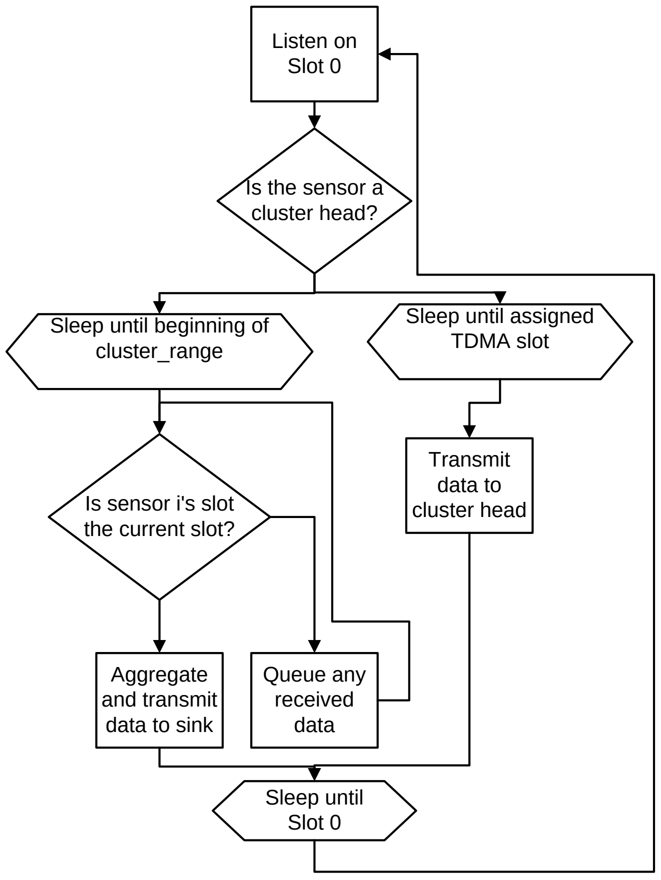
Listen on Slot 0
Is the sensor a cluster head?
Sleep until beginning of cluster_range
Sleep until assigned TDMA slot
Is sensor i's slot the current slot?
Transmit data to cluster head
Aggregate and transmit data to sink
Queue any received data
Sleep until Slot 0

Figure 7 - State-State Phase

4.2.1 TDMA Scheduling

CEMS employs a global TDMA schedule (i.e. all sensors and clusters participate) to manage channel access among sensors and the base station. There is a single broadcast slot at the beginning of each cycle, while the remaining slots are strictly unicast. The CEMS TDMA scheme is shown in Figure 8.

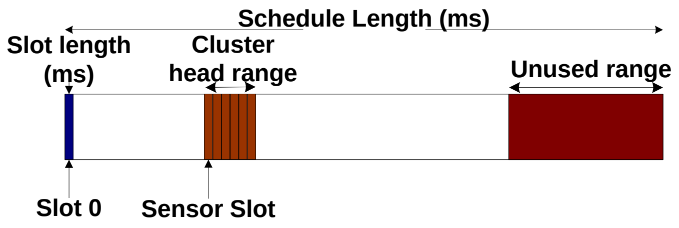

Figure 8 - TDMA Schedule

Each sensor is given a unique slot in the TDMA schedule during each reclustering phase. Note that there is no guarantee a sensor's slot will be the same in two different rounds of operation.

All sensors, including cluster heads, transmit to the base station on their slot. All nodes must also listen on slot 0, which is reserved for broadcast communications from the base station. Furthermore, cluster heads must listen during each slot in their cluster's range to receive data from their members. To minimize hardware delays resulting from switching between sleep and wake states, slots within a single cluster always form a contiguous block of slots. Sensors are in sleep mode at all other times, their radio electronics turned completely off.

4.2.1.1 Dynamic and Static Schedules

On the surface, dynamic TDMA schedules are much more flexible and able to adapt to changing network conditions than static schedules. Delay can be minimized, and medium use can be better allocated as the sensor population changes. Despite these benefits, we chose to implement a static TDMA schedule.

The decision to do so was based on two factors. From an implementation standpoint, hardware support for dynamically resizing TDMA cycles is limited and would restrict deployment of CEMS to a smaller number of platforms than would otherwise be possible. This is a problem experienced by LEACH and LEACH-C, which rely on intra-cluster TDMA schedules with a number of slots equal to the current number of members.

A more subtle problem with a dynamically resizing TDMA schedule in response to decreasing numbers of sensors must also be addressed. Changing the number of slots in a schedule to always correspond to the current sensor population has the effect of changing the rate at which data is reported as sensors begin to die, since cycles will repeat more or less often as sensors are removed or added to the population. This not only results in a variable measurement rate, but also serves to hasten network death by causing more transmissions over time. Thus, CEMS chooses not to resize the global TDMA schedule in response to sensor death.

4.2.1.2 Scalability and Delay

A common objection to TDMA, especially when implemented globally as CEMS does, is that the network has trouble scaling to high sensor populations. On the surface this is true; large numbers of sensors necessitate proportionally large TDMA schedules and therefore low per-sensor data rates and significant delay. Despite this, global TDMA networks are scalable for many applications. A given TDMA slot may have a length of 50ms or less, depending on the nature of the data reported (e.g. temperature data may take much less time to transmit than a video image). Assuming that a network initially has 100 sensors, one TDMA cycle lasts approximately 5 seconds. Cycle lengths scale linearly with sensor population, so doubling the number of sensors would increase the delay to 10 seconds.

Many applications of sensor networks can tolerate significant amounts of delay, such as soil chemistry sensors or building stress monitors. In these and similar scenarios, values being reported once per minute is more than sufficient. Under such constraints a network could be scaled up to 1200 sensors and still meet minimum delay requirements. However, query-based networks or real-time object-tracking networks may not be suitable applications for a CEMS-based system.

4.2.1.3 Collision Avoidance

Global TDMA address several problems inherent in WSNs and wireless networks in general. Dividing the medium into a series of equal-length time slots ensures that at no point will more than one sensor be transmitting over the wireless medium. Collisions are impossible under global TDMA, ensuring that no energy-intensive retransmissions due to congestion or collisions will take place. Furthermore, only the base station and one cluster head are capable of hearing transmissions at a given time (with the exception of slot 0). Energy losses due to overhearing, which can be a significant drain on sensor energy levels in dense network deployments [33], are thus avoided during the steady-state phase.

Finally, CEMS avoids potential hidden terminal problems among cluster heads. If a base station is centrally positioned in the middle of a field, sensors' effective transmission and reception ranges may not extend to all edges of the field. This introduces the possibility of collision among transmissions meant for the sink, which is a problem even in clustering protocols like LEACH-C which use CDMA (Channel Division Multiple Access) spreading codes to prevent interference among adjacent nodes and CSMA (Carrier Sense Multiple Access) to handle cluster head communication. CSMA checks to see if any transmissions are ongoing in the medium before a node tries to initiate its own transmission. If the medium is clear, it broadcasts. If not, it backs off for a period of time and tries again. Cluster heads must transmit on the same channel that the base station is listening on (precluding CDMA), and therefore may interfere with one another in certain circumstances despite their use of CSMA.

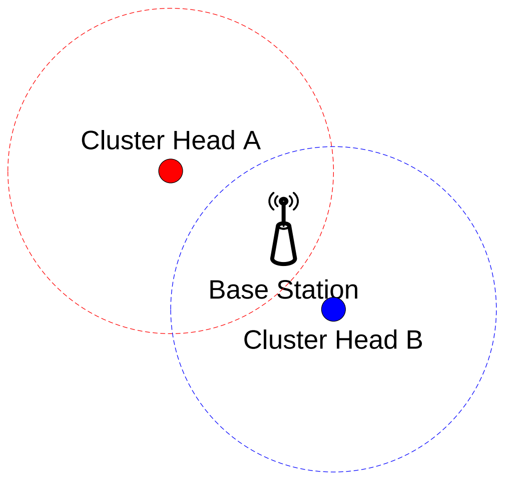

Figure 9 - Hidden Terminals

Consider the situation illustrated in Figure 9. Cluster heads A and B are both within range of the base station, but not of each other. If Cluster head A senses the medium prior to transmitting to the sink, it will appear to be free despite the fact that Cluster head B may be actively transmitting. If Cluster head A then begins its own transmission, they will interfere at the base station and prompt both cluster heads to retransmit.

4.2.1.4 Synchronization

Due to time constraints, CEMS does not explicitly model any synchronization mechanisms or simulate clock drift. The protocol is designed, however, to easily accommodate timing beacons broadcast by the sink during slot 0 of each TDMA cycle. If the underlying hardware platform supports it, physical-layer reference broadcast synchronization would allow for a low-energy solution to clock drift. Otherwise a short MAC-layer reference beacon frame could be easily implemented into the CEMS protocol.

4.2.2 Quick Recovery

While clustering does reduce the energy load on wireless sensors, it also introduces single points of failure for each cluster. In many real-world environments sensors may fail due to, e.g., vandalism, theft, or environmental effects. Loss of a cluster head not only removes that sensor from the network, but it also breaks the routing path for all sensors in that cluster. This could significantly impact network coverage in the affected area, which for some WSN applications represents failure of the network. Given this dependence on certain individual sensors, it is somewhat surprising that existing clustering protocols do not handle cluster head loss elegantly. At best the cluster might be offline until the next round of reclustering. At worst, the entire cluster may be declared dead and removed from the network.

4.2.2.1 Coverage Loss

Precisely how significant the loss of a cluster head is to a network in terms of coverage depends on sensor density, the number of clusters in the network, sensor failure detection mechanisms, and network topology. Areas of high sensor density tend to experience overlapping areas of coverage for many sensor applications. Unfortunately, networks are not necessarily uniformly dense, and redundant coverage areas may be present only by chance. Networks that deliberately implement large areas of redundant coverage may do so due to the necessity of ensuring accurate data in exchange for a higher initial investment in sensor nodes or modules. These networks are resilient against individual node failure, but may not be able to compensate well for cluster head loss.

As the authors of [2] have analytically shown, the number of clusters (and thereby cluster heads) is small relative to the total number of sensors for many applications. While this is useful from the standpoint of energy efficiency, it also places a great deal of importance on a small number of sensors. Loss of cluster head when there are only a few clusters total has the potential to bring large fractions of a network offline, opening significant holes in its coverage area. Even redundant networks may be vulnerable to this effect, as spatially adjacent (and therefore overlapping) sensors tend to be members of the same cluster.

Coverage loss may be exacerbated by the mechanism a protocol uses to detect failed sensors. If a sensor is unable to communicate with the sink due to its cluster head being offline, it is possible that the base station will assume that the sensor has died and remove it from future clusters. If such is the case, loss of a cluster head may permanently remove all cluster members from the network. This is less likely in a distributed system like LEACH, but centralized protocols such as HCR or LEACH-C may be vulnerable.

Finally, networks which employ super-clusters or cluster-head backbones may experience coverage loss from multiple clusters if an upstream node fails and their routing paths are cut. These networks often employ specialized nodes equipped with energy harvesters to avoid just such a fate, but the sensors may still be damaged or removed due to environmental conditions or deliberate vandalism.

4.2.2.2 Failure Detection and Emergency Reclustering

CEMS uses the periodic nature of its global TDMA cycle to rapidly recover from coverage loss due to cluster head death. At the beginning of each steady-state phase, the base station computes the expected transmission times of each cluster head using its TDMA slot and the overall cycle length. If any cluster head fails to transmit during its expected time, the sink increments that sensor's missed transmission count. Three missed transmissions result in that sensor being labeled as dead, and trigger an emergency reclustering event to reconnect the cluster to the WSN. A successfully received transmission resets the sensor's missed transmission count.

Note that a tradeoff exists between the recovery period and accurate classification of cluster head death. The more missed transmissions required before a sensor is declared dead, the longer a cluster may be offline before emergency reclustering is triggered. A small missed transmission count, however, is vulnerable to false positives. In the field a sensor's transmissions may be blocked by a mobile obstacle (e.g. a passing vehicle), interfered with by a spike in radio noise, etc. Misinterpretation of these temporary problems as permanent sensor death could lead to unnecessary energy expenditure and downtime due to reclustering.

4.3 Configuration

A number of CEMS's parameters can be configured to meet application-specific constraints. In many sensor applications, for example, significant amounts of delay are acceptable if the network itself will last longer as a result. The density of sensors and the importance of continual coverage is also a factor in network lifetime. Finally, the optimal number of clusters to partition the network into may vary somewhat based on the previous considerations.

A critical metric in our simulations is network lifetime, a measure of the length of time during which a network can be considered capable of monitoring its environment. Specific definitions of network lifetime have been described differently in different publications, and depend on the use to which a sensor network is put. For some applications the death of a single sensor may be defined as network death, while for others a more relaxed constraint may be employed. Any application-specific configuration of CEMS will depend heavily on how this metric is defined.

4.3.1 TDMA Schedule Length

The minimum length of CEMS' TDMA schedule is somewhat dependent on the kind of data being sensed by the network. Each slot needs to be at least long enough to accommodate a sensor's radio electronics' wakeup time and the transmission of sensor data to the cluster head or sink. Note that wakeup times are constant, however, and can be accounted for by starting the wakeup process during the previous slot. Beyond this minimum, extra slots can easily be added at the end of the TDMA cycle. While slot lengths could also be increased, this would result in sensors staying awake for longer periods and expending unnecessary energy on their transceiver modules. Adding empty slots, however, introduces periods where all sensors are asleep. This reduces the frequency of sensor reports, allowing fewer radio transmissions and thereby reducing energy costs.

Note that this configuration is most useful where the minimum sensing granularity is significantly better than required. A network monitoring soil pH, for example, may not need a granularity higher than 1 measurement per sensor per hour. In this case a great deal of delay can be added to the TDMA cycle, dramatically increasing network lifetime without violating the latency constraints imposed by the application.

4.3.2 Reclustering Period

The ideal duration of each reclustering period in CEMS is application-specific. Figures Figure 10 and Figure 11 show the tradeoff between coverage and lifetime for reclustering periods of 20 hours and 100 hours, respectively. Each graph shows the residual energy over time for each sensor in the network. Given an initial population of one hundred sensors, the simulation begins with five clusters. Sharp declines in energy correspond to being made a cluster head, while gradual energy loss represents cluster membership.

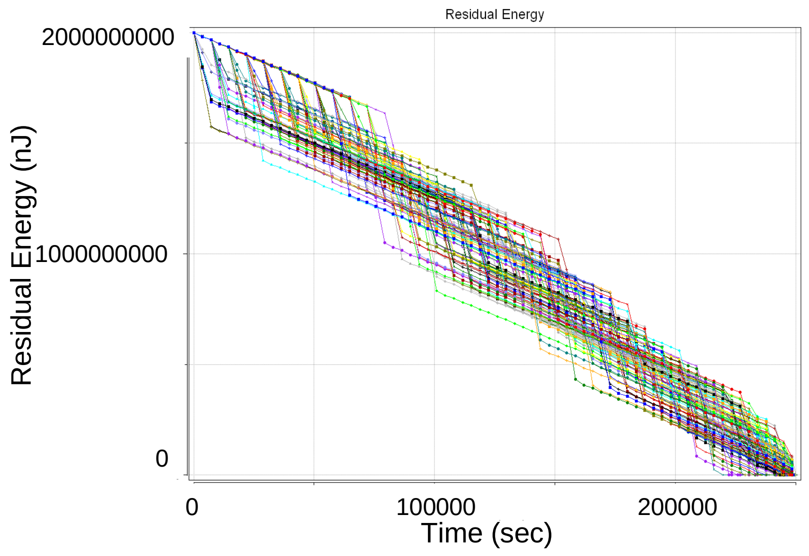

Figure 10 - 20 Hour Reclustering Period

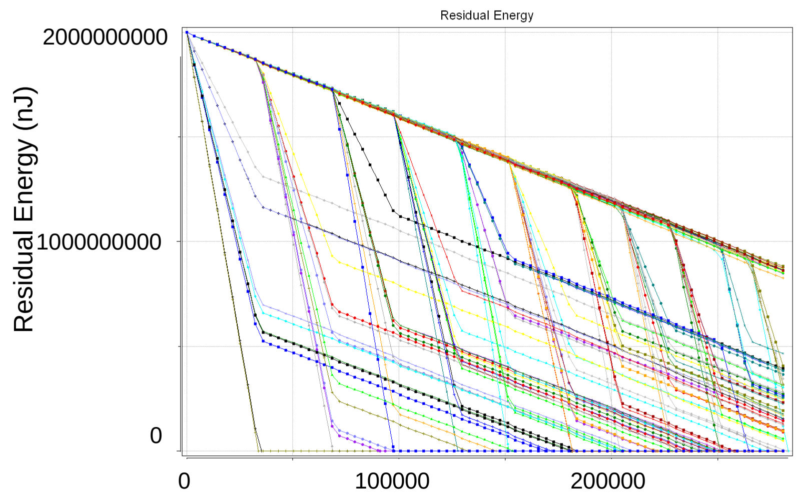

Figure 11 - 100 Hour Reclustering Period

The relatively narrow gap between the sensor with the lowest residual energy and that with the highest residual energy in Figure 10 indicates a fairly even balancing of energy costs over the network. This preserves coverage for as long as possible, after which all sensors die within a few hours of each other. Figure 11, conversely, begins to lose sensors almost immediately. In this configuration, cluster heads lost so much energy before being reassigned as cluster members that they die shortly after reclustering. Note, however, that while coverage is significantly worse than the previous case, the overall network lifetime is extended by approximately 12%. Figure 12 illustrates the relationship between these two factors.

Therefore, in dense networks with overlapping areas of coverage or for networks which do not require complete coverage, a long reclustering period may be preferable. For networks where coverage must be maintained for as long as possible, a shorter reclustering period is desirable. For our experiments, we chose to use a reclustering period of 20 hours.

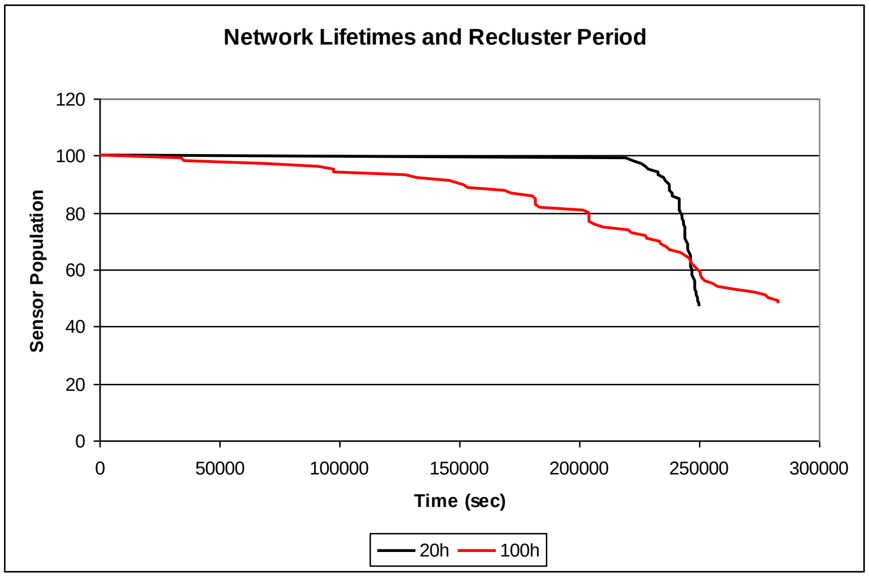

Figure 12- Reclustering Period and Network Lifetimes

4.3.3 Cluster Size

Empirical testing as well as analytic analysis presented in [2] indicates that a number of cluster heads equal to 5% of the original population provides the longest network lifespan when most sensors are expected to die near the end of the network's operative span. In applications where sensor death is more evenly distributed over the life of the WSN, however, a number of clusters equal to 10% of the current population allows for smooth scaling of cluster sizes. This scenario occurs most often when reclustering rates are high, and there is sufficient sensor overlap to sacrifice individual coverage of sensor nodes (e.g., the 100h period in Figure 12). When the reclustering period is high and/or the network is dense, more clusters might be desirable to allow for tighter groupings as sensors regularly fail. Since CEMS is primarily investigated in the context of failure-prone networks, our experiments use 10% of the current population as cluster heads.

4.3.4 Radio Model

In order to establish a fair basis of comparison with LEACH-C CEMS uses an equivalent radio model (see Section 5.1.2). Transmissions of one meter or less are modeled using a free space radio model, while transmissions beyond one meter use model closer to two-ray ground reflection with an α=4 power loss exponent:

$$\text{If } d \le 1: E_{Tx}(k,d) = (E_{elec} * k) + (\varepsilon_{amp} * k * d^2)$$
$$\text{Else}: E_{Tx}(k,d) = (E_{elec} * k) + (\varepsilon_{amp} * k * d^4)$$
$$E_{Rx}(k) = E_{elec} * k$$

- E_{tx} – Energy required to transmit
- k – Length of the message to be transmitted
- d – Distance to transmit
- E_{elec} – Radio electronics' energy dissipation in nJ/bit
- ε_{amp} – Energy dissipation of transmit amplifier in pJ/bit/m^2

This model, while not realistic in the sense that multipath and shadow fading are not addressed, does represent the capabilities of common sensor transceivers in terms of energy costs for transmission and reception. Given that neither CEMS nor LEACH implements any techniques that are designed to mitigate signal attenuation, it is not unreasonable to use the above as a radio model for simulaton.

5 Simulations

Given the plethora of existing clustering protocols for wireless sensor networks, performance analysis of CEMS focused on its ability to quickly restore coverage in fault tolerant environments and to prevent interference amongst nearby sensors. To this end, a detailed model of the Media Access Control and physical layers was necessary to accurately gauge performance in a realistic setting. To this end we elected to implement CEMS in v3.2 of the Omnet++ discrete event simulator [10] and its Mobility Framework module[11], which provides detailed modeling of the lower levels of the network stack. Due to time constraints, our version of LEACH-C is the same as used in [3].

Ideally, these simulations would have modeled the wide variety of accidents and problems that are encountered in real-world settings, such as noisy radio channels, mobile obstacles causing intermittent effects on the wireless medium, and the reflection, diffraction, and scattering of radio transmissions off of both mobile and static objects. Unfortunately, the sheer complexity of modeling radio propagation on that level is impossible without access to significant computing resources. Simulation modules would have to be specially developed for Omnet++ that implemented such modeling, which was impossible given the CEMS development time constraints . We instead elected to simulate sensor death without its cause. To this end we implemented a Poisson model to trigger sensor failure, detailed below.

5.1 Software Tools

5.1.1 Omnet++ and the Mobility Framework

Omnet++ is a powerful discrete event simulator used in a number of wireless mobile, ad-hoc, and sensor network simulations. A variety of supported frameworks such as MiXiM (used by the authors of Crankshaft) and the Mobility Framework extend the capabilities of the base simulator with detailed models of lower-layer protocols and physical layer modeling of the wireless medium. For our purposes, the Mobility Framework sufficed to install an 802.15.4 Zigbee MAC layer over which CEMS operated as a Layer-3 protocol.

Unfortunately, this simulator is much more popular in Europe than in America. Core documentation, such as the Omnet++ manual and an introductory tutorial, are well written overall. However, a number of problems that users are likely to experience with both the base simulator and any frameworks are not addressed in the core documentation. This is compounded by error messages that, while meaningful to developers, are often unclear in their intent to the user. Support is only available through a mailing list which is subscribed to by a number of other users, and a few developers.

These circumstances can easily lead to situations in which an unknown bug (or design decision not supported by the simulator) causes an error message that seems unrelated to any problems actually being encountered, and can only be resolved by posting a question to developers who already receive many daily requests for assistance. Turnover time for questions can occasionally take days, during which development of the user’s simulation could be effectively halted.

The Mobility Framework (MF) is especially vulnerable to these kinds of problems, as bugs have been discovered in the included modules which only arise in very specific and infrequent circumstances. When these bugs do manifest, however, they are often difficult to diagnose and significant enough to halt a running simulation. Furthermore, the MF's manual is outdated to the point of being incorrect. A major change to system architecture has been made since the manual was first published. This leaves the user with little more than the API for documentation, and not all classes, functions, or data structures are actually described in any detail within the API.

5.1.2 LEACH-C Implementation

We compare CEMS to the LEACH-C protocol described in [3]**.** LEACH-C is a centralized version of the original LEACH [1], replacing its randomized rotation system with a simulated annealing algorithm running on the base station. During LEACH-C's cluster formation phase, each node transmits position and energy data to the base station. It disqualifies any node whose residual energy is less than the mean node energy, and includes the remainder as a set of objects to be input into a simulated annealing algorithm. Solutions are evaluated using a minimizing scoring function similar to that used in k-means. Each cluster is scored based on the sum of the squared distances between each cluster member and the cluster head.) Note that the number of clusters is determined *a priori*.) Once a good solution is found, nodes are informed of their new status and the same steady-state phase that LEACH employs is started.

In LEACH-C sensors communicate with cluster heads using a local (i.e. cluster-specific) TDMA schedule. Since cluster sizes may change during each reclustering, a dynamically resizing TDMA schedule whose operational details are assumed to be handled automatically by the sensor is used. This is difficult to implement on a hardware level, and may limit the number of sensors compatible with the protocol. Each cluster uses a different CDMA spreading code to avoid interference, and cluster heads transmit to the base station using a contention-based CSMA system. While this causes less delay than CEMS' global TDMA schedule, hidden terminals among cluster heads may still be a problem if sensors do not have radios powerful enough to sense the entire field before deciding to transmit.

LEACH-C uses a free space radio model for transmissions under one meter. For longer distances, the authors state that a multipath model is used. In reality the radio model resembles two-ray ground reflection model with an α=4 power loss exponent. No actual modeling of multipath fading or shadow fading appears to be implemented. Since we compare against the version of LEACH-C implemented in [3], however, it is possible that the original model written in ns-2 did employ a more advanced simulation of radio wave propagation. CEMS employs an equivalent model to that used in [3] in order to ensure a fair comparison.

5.2 Assumptions

Our experiments were governed by several key assumptions. One of these is imposed by assumptions made in [3], while the remainder are based on common real-world configurations or made in order to limit the scope of CEMS to a reasonable level given the time constraints placed on its development:

- **There is no radio background noise or interference**: While not realistic for most settings, no WSN simulator that we investigated had sufficiently detailed physical-layer modules to accurately portray radio wave propagation in any detail. Furthermore, CEMS is designed to operate above a WSN MAC layer. Ensuring that frames arrive at their destination is not currently part of the protocol’s responsibilities. Note, however, that some modifications might be necessary if background noise is sufficient to delay successful transmission beyond the end of a sensor’s TDMA slot.
- **There is no spontaneous packet loss**: Similar to the above assumption, we assume that only signal attenuation affects packets in transit over the wireless medium. Packets once sent will always arrive, though at a lower signal strength.
- **All sensors are initially homogenous**: At the beginning of a network’s lifespan, all sensors have the same amount of residual energy. All nodes are therefore equally likely to be considered for cluster head status, and position will be the sole determinant of which sensors serve as cluster heads in the first round.
- **Sensor platforms are homogenous**: All sensors have the same underlying hardware. No node is equipped with energy harvesting modules or long-range transceivers, and only the sink is not equally energy constrained.
- **All sensors can transmit to the base station**: Node’s radios are always sufficiently powerful to reach the base station, located in the center of the field for all but one of our experiments. Every sensor is therefore capable of being a cluster

head. Note, however, that this assumption does not imply that all sensors can transmit to all other sensors (see Section 4.2.1.3).

- **Sensor locations are static**: No sensor node is mobile. Once placed a sensor remains at its initial position throughout the simulation. CEMS is not designed as a protocol for mobile WSNs and would not be an appropriate choice for such a network. Mobility is an active research area in and of itself, with a variety of specially designed protocols and routing mechanisms that exceed the scope of this project.
- **There is no clock drift**: Simulating the effect of clock drift on a TDMA-based was deemed too complex of a problem to handle in the time allotted to CEMS development. A discussion of this problem is presented in Section 4.2.1.4, however.
- **Any node, including a cluster head, has a non-zero probability of failing at a given time due to random accidents**: In real-world environments, sensors are vulnerable to a number of disabling situations. Vandalism, storm damage, unintentional destruction, removal from the field, or permanent signal blocking are all possible fates for a wireless sensor.
- **Cluster heads are capable of perfect compression/aggregation**: This is an assumption made by [3]. A cluster head is able to process any number of messages from cluster members in such a way that its transmission to the base station contains the same amount of data as a single sensor's transmission to the cluster head. This effectively decouples the size of reported data from the size of a cluster, and is rather unrealistic for many applications. The result of 100 sensor reports, for example, should not be the same length as the result from 1 sensor report unless some extreme form of data filtering (such as only reporting the highest sensed value) is in use.

5.3 *Poisson Death Model*

A network-wide death model using Poisson interarrival times is used to simulate accidents happening to sensors in the field. The cause of these deaths is not addressed, but it is assumed that each accident is entirely fatal. No sensor will ever be damaged but operational; a node is either functioning or dead. Each arrival generated by the model is treated as a sensor death. The specific sensor is determined randomly using a uniform distribution. This could easily be changed, however, to reflect the existence of high-risk sensors.

Days were chosen as a unit with which to represent expected arrival times of a sensor death. Note, however, that this is a fairly arbitrary choice based on desired granularity; a Poisson model is unit-less. Minutes or hours could just as easily be used, though this would require adjusting the expected value of deaths per unit time.

5.4 Field Description

One hundred sensors were scattered over a 25m square field. A uniform distribution was used to assign spatial coordinates, and the base station was placed in the center of the field. Given the uneven placement of sensors in relation to the field, certain regions of the network are extremely dense (e.g. the region between (3, 21) and (5, 23) in Figure 13) while others are quite sparse (e.g. the region between (20, 14) and (24, 13) in Figure 13.

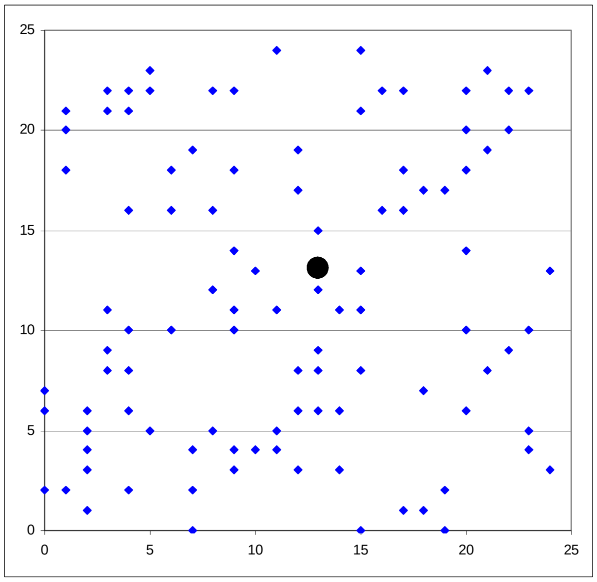

Figure 13 - Sensor Distribution

Previous discussions have examined the impact of network density on CEMS configuration. Since the experimental network is composed of both dense and sparse regions, CEMS is not tuned to provide optimal quality of service for either extreme. Coverage in this network obviously does not include the entire field. Any analysis of coverage, then, is compared against the initial amount of the field that is monitored at the start of the network's lifespan.

6 Experiments

Given the plethora of existing clustering protocols for wireless sensor networks, performance analysis of CEMS focused on its ability to quickly restore coverage in fault tolerant environments and to prevent interference amongst nearby sensors. A number of other useful experiments are certainly possible, but time constraints dictated that the majority of experiments relate directly to CEMS' function as a fault-tolerant clustering protocol. Potential other experiments are discussed in Section 7.

6.1 Network Lifetime

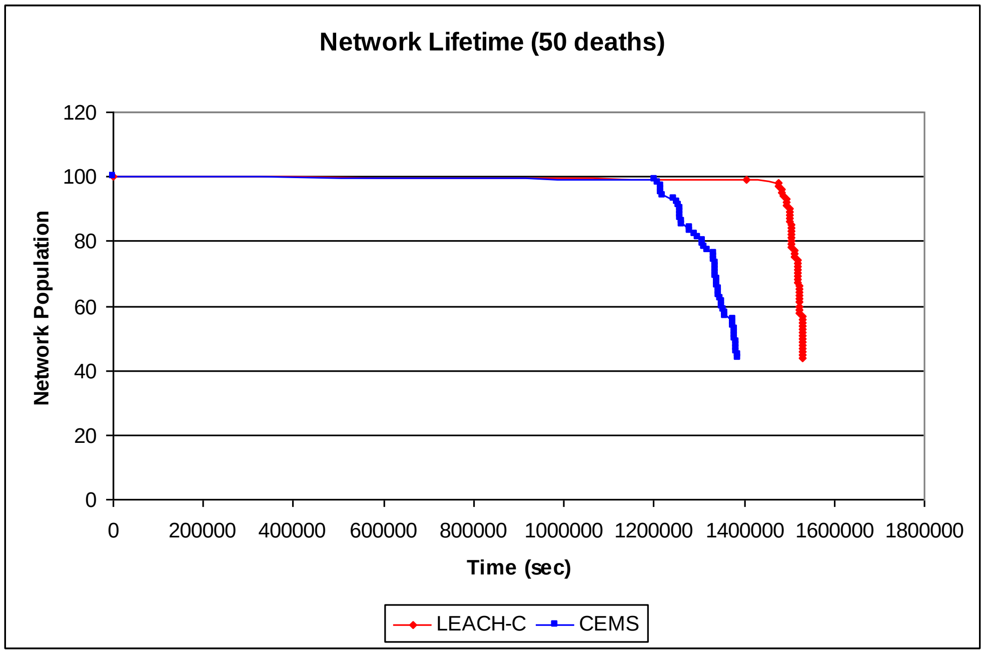

Figure 14 - Network Lifetime

Using 2J of energy and no random accidents, CEMS and LEACH-C were both run until network death (defined initially as the loss of 50% of the original population). As can be seen in Figure 14, LEACH-C currently lasts approximately 8% longer than CEMS. Both maintain fairly consistent coverage until the end of their operative lifespans, at which point all sensors die within a short time interval. Since each protocol's underlying radio model is equivalent, the disparity is lifetime is due to aspects of each system's design. Data packet sizes dwarf the energy costs associated with control packets[4], and neither LEACH-C nor CEMS is subject to overhearing the transmissions of neighbors (due to CDMA codes and global TDMA, respectively).

The only significant difference between the two protocols which affects energy consumption, therefore, is the method of arriving at clustering solutions. CEMS employs a genetic algorithm which scores potential solutions based on the physical proximity of sensors, while LEACH-C's simulated annealing algorithm uses a combination of energy levels and spatial locations of sensors. Time constraints prevented a comparison of the two protocols using the same clustering algorithm, but there is no reason that such a study couldn't be made. The essential functionality of CEMS, its quick recovery behavior and global TDMA schedule, are not coupled to the choice of clustering algorithm. This possibility and the performance implications of using a different technique are investigated below, in Section 6.4.

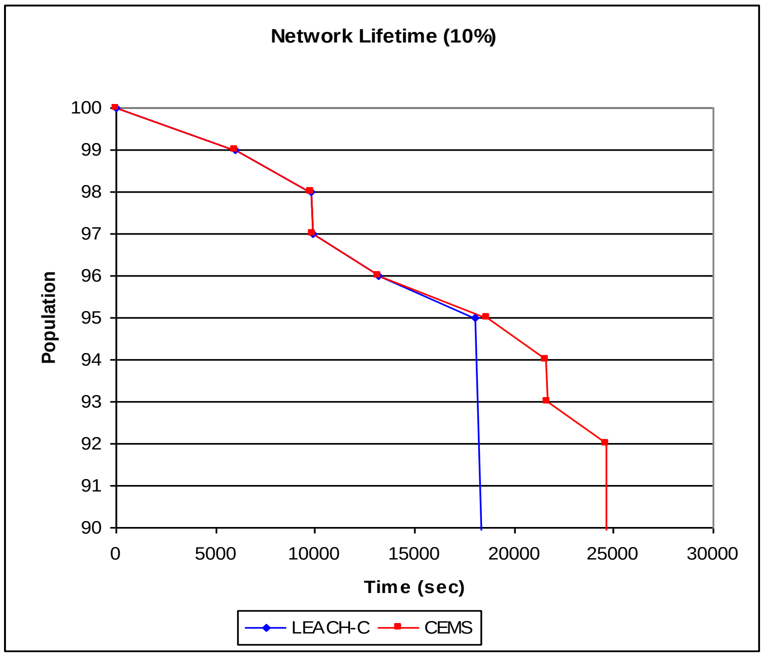

Figure 15 - Network Lifetime = 10% of the original population

[4] Data packets are 500 bytes (see Appendix I), while control packets are 30 bytes or less.

To investigate the impact of accidental sensor death on network lifetime, we increased the rate of random deaths to an expected value of five sensors per day. Figure 15-17 show the effect of sensor death at lifetimes defined as 10% of the population, 25%, and 50%, respectively. In order to clearly show actual sensor populations, the populations, the temporary loss in network population due to cluster head death is investigated separately in the context of network coverage, below.

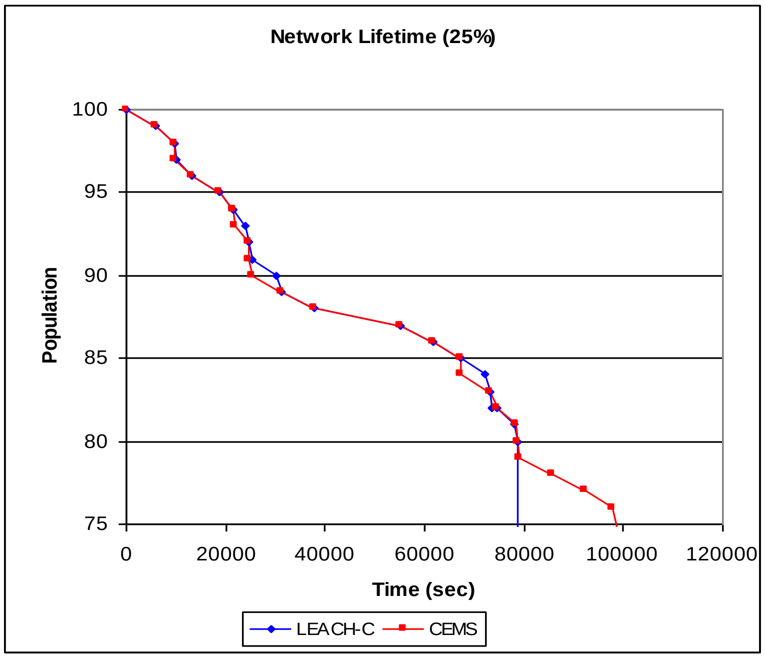

Figure 16 - Network Lifetime = 25% of the original population

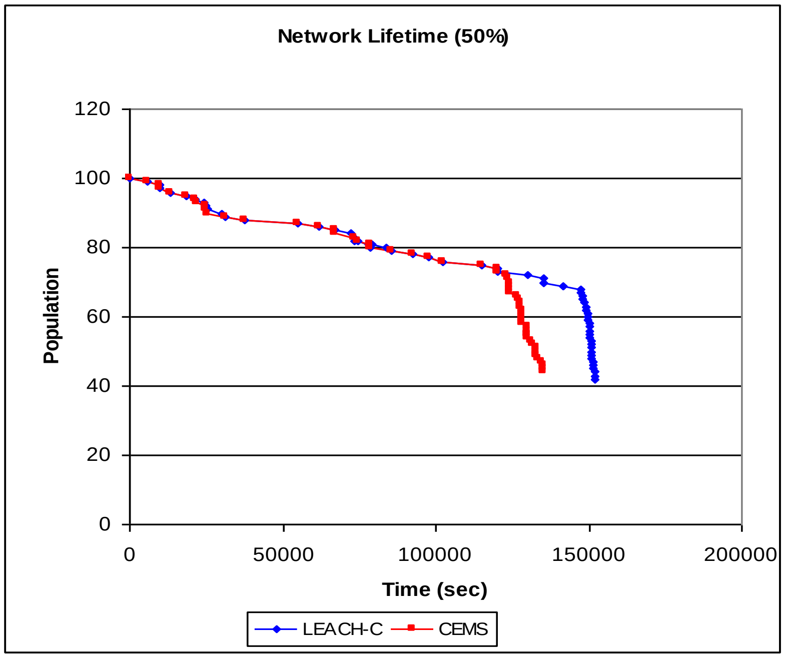

Figure 17 - Network Lifetime = 50% of the original population

Figures 15-17 display the actual sensor population, including temporarily offline but still living sensors, though network death may still be caused by clusters going offline due to cluster head death.

As can be seen in Figure 15 and Figure 16, CEMS remains operational for approximately 26% and 20% longer than LEACH-C, respectively. This is due to the impact of cluster head death on a network when cluster sizes are small. The loss of a cluster head in LEACH-C may put 20% of the sensors offline until the next reclustering period. If this puts the network below its minimum acceptable number of living sensors, the network is unable to fulfill its role and may be considered dead. In Figure 17 the LEACH-C network remains operational approximately 8% longer than CEMS. Since the network may lose up to 50% of its sensors before being considered dead, temporary sensor losses due to cluster head death are less significant. There may be significant effects on network coverage, however. Except in the case of very dense networks, losing almost half of all sensors in the field may introduce holes in the area that can be effectively sensed.

6.2 Coverage

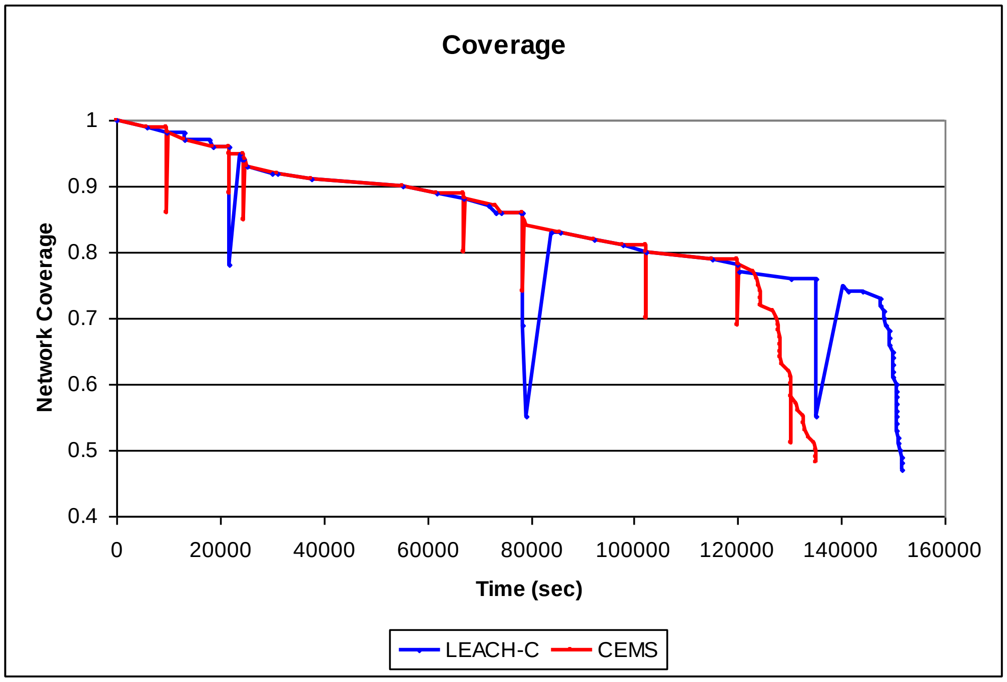

Figure 18 - Network Coverage

For this experiment, sensor failure rates were again set to an expected value of five sensors per day. Despite the shorter overall lifespan of CEMS at a network lifetime of 50%, Figure 18 demonstrates its ability to retain a high standard of network coverage in the face of cluster head death and high sensor failure rates. As can be seen in Figure 18, both networks experience sensor death at the same points in time. Those deaths that are merely cluster members create a small drop in the network's overall coverage. Larger drops are caused by the loss of a cluster head, which destroys the ability of all sensors in that cluster to transmit their information to the sink. Since a given sensor may not be assigned as a cluster head in both networks, the impact of a specific sensor death event on network coverage may be more or less pronounced (as can be seen in Figure 18). Since LEACH-C creates a number of clusters equal to 5% of the current sensor population, the loss of a cluster head is also more significant in terms of coverage loss than in CEMS, which uses 10% of the current population to determine cluster size. However, the probability of cluster head loss is also proportionally lower. Note that CEMS recovers from cluster head death rapidly by reclustering in response to missed transmissions from cluster heads. LEACH-C, conversely, does not restore coverage until the next periodic reclustering is triggered. This not only results in potentially significant gaps in the sensed area, but increases the probability of multiple clusters being offline at once. This is evident at approximately t=80000 in Figure 8.

6.3 Sink Location

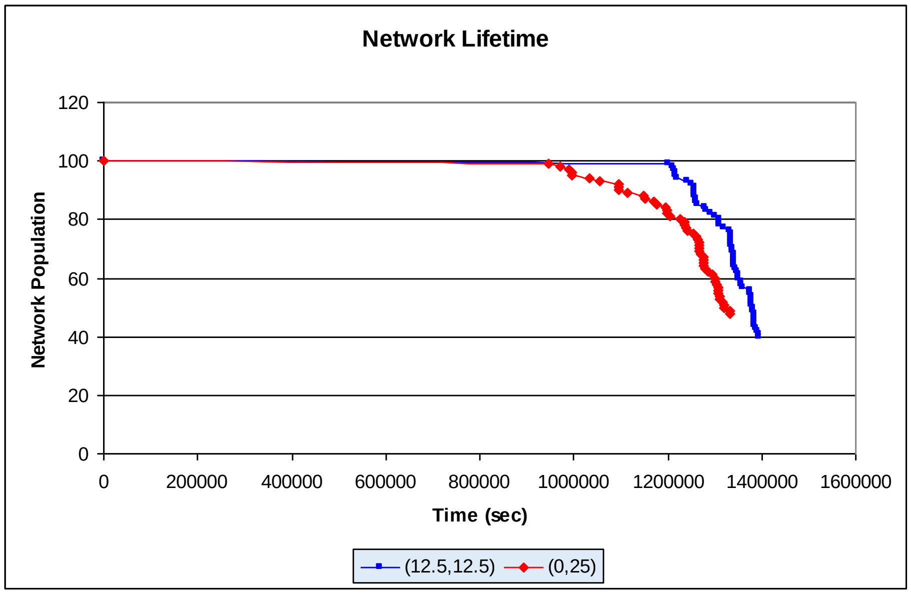

Figure 19 – CEMS Algorithm with Varied Sink Location

For many WSN applications, the sink cannot be placed in the center of the field. To investigate the impact of sink location on network lifetime, we compared CEMS' standard base station placement in the center of the field to one in which the sink was moved to a corner. Each sensor was given 2J of energy and placed in a 25m square field as described above. In the reference simulation, the sink was placed at (12.5, 12.5). In the experimental simulation, the sink was placed at (0, 25), the upper-left corner of the field.

As can be seen in Figure 19, moving the sink reduced network lifetime by approximately 5%. In addition, sensor death times were spread over a larger interval than in the reference simulation. This has the effect of reducing network coverage earlier in the network's operative lifespan.

Unsurprisingly, sensors died at a rate commensurate with their distance from the base station. Figure 20 illustrates this phenomenon. The chart duplicates the sensors' positions in the field as in Figure 13, the sink is moved to (0, 25) and but each sensor is colored to show the time it was observed to fail. At t=1100000, only sensors on the far side of the field had failed due to lack of residual energy. By t=12000000, a wider band of sensors closer to the sink were also dead due to due to energy dissipation. By the end of the simulation, only those sensors closest to the sink remained active.

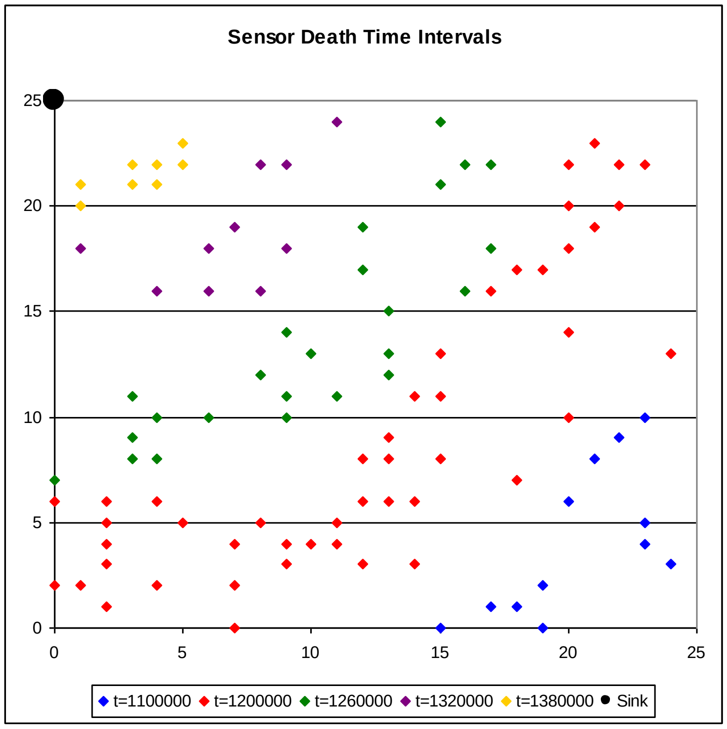

Figure 20 - Sink Location and Sensor Death

Interestingly, cluster heads do not expire significantly before their cluster members. While increasing the granularity of the results would certainly indicate cluster head failure preceding cluster member failure by some extent, the time difference is not large enough to actually register at normal sampling rates. This indicates that energy load is being balanced fairly effectively.

This experiment does suggest a potential improvement to CEMS, however. In situations where cluster to sink transmission differences vary widely over the network (e.g. in the above example or in large fields), the protocol might be modified to rotate cluster heads during the steady-state phase at rates dependent on the cluster's mean energy dissipation over time. This would preserve cluster memberships while further balancing energy load.

6.4 Clustering Algorithm Comparison

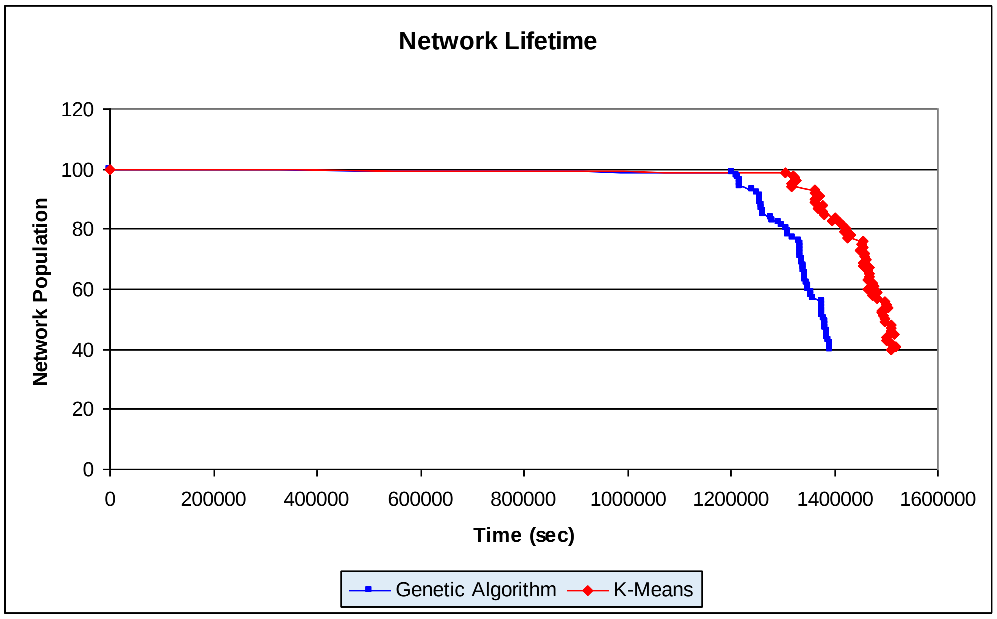

Figure 21 - Clustering Algorithm Comparison

As discussed in Sections 4.1.1.2.1 and 0, a genetic algorithm may not be the most appropriate clustering technique given that CEMS clustering assignments are based only on spatial proximity. Since the protocol is decoupled from its clustering algorithm, we tested CEMS using both its original GA and a k-means algorithm which uses iterative refinement to partition a network into cluster domains. The head selection algorithm described in Section 4.1.2 is used in both experiments to elect cluster heads. The version of k-means that was employed in this experiment uses the kmlocal [34] library developed by the University of Maryland.

As can be seen in Figure 21, k-means's clustering solutions result in an extended network lifetime of approximately 8%. This makes CEMS networks perform almost equivalently to LEACH-C networks in terms of network lifetime in normal conditions. Coverage loss is similar to that experienced when using the genetic algorithm. Furthermore, the k-means algorithm executes with a mean time of 1.42 seconds, as compared to the genetic algorithm's mean run time of 15.38 seconds. This significantly decreases the time required to simulate network operation, and would be valuable in real-world conditions to decrease the delay associated with emergency reclustering. Had this experiment been run earlier during CEMS' development, k-means would have been employed instead of the currently implemented genetic algorithm.

7 Conclusions

Several conclusions can be draw from these simulation experiments. First, there is an important application-specific tradeoff between acceptable coverage and network lifetime. By reclustering infrequently, a network may have a longer operative lifespan at the expense of early and increasingly common gaps in its coverage. For denser networks or those monitoring conditions likely to register on multiple sensors, this may be an acceptable tradeoff. For sparser of more precise networks, however, a decreased lifespan may be an acceptable cost for ensuring good coverage. A further tradeoff must be made between the number of clusters in a network and the expected failure rate of sensors due to accidents. A small number of cluster heads cause a significant loss of coverage if they fail, while a larger number of cluster heads cause a proportionally smaller coverage loss. Assigning more sensors to the role of cluster head also increases the probability of a cluster head randomly failing, of course.

Secondly, a synchronized global TDMA schedule allows the base station to predict when transmissions from given cluster heads are expected and ensures that no sensors will act as hidden terminals. CEMS uses the former feature to implement a quick recovery system that rapidly restores network coverage in the event of cluster head death.

Third, while LEACH-C has a longer operative duration than CEMS under normal conditions and loose definitions of network lifetime, it cannot maintain a high degree of coverage under failure-prone conditions or stricter lifespan requirements. A potential avenue for future research might be the adaptation of a quick-recovery mechanism to the LEACH-C protocol, however.

Fourth, the location of a WSN base station relative to its sensor nodes has a potentially significant impact on network coverage. Since CEMS does not currently support cluster head rotation during steady-state mode, sensors and clusters farther from the base station fail due to low residual energy earlier in a network's lifespan than those nearer the base station. This is especially obvious in large fields and topologies in which the sink is not centrally placed.

Finally, a k-means algorithm is more appropriate for cluster selection than a genetic algorithm when only spatial proximity is considered. While GAs are popular methods of selecting cluster members, the benefits of evolutionary searching are somewhat lost when a single metric should be minimized or maximized. An evaluation of the performance of single-metric clustering techniques to more complex algorithms which consider many factors (e.g. position, energy, projected transmission costs, etc.) might be an interesting study, however.

CEMS contributes to the study of wireless sensor networks by exploring the behavior and performance of WSNs in failure-prone conditions, and addressing the problem of coverage loss via its quick recovery mechanism. The majority of existing research does not address the problem of random sensor failure, which in any cluster-based network can pose significant risks. The benefits of a synchronized, global TDMA schedules for delay-tolerant applications are also addressed in this project. The ability to predict sensor activities in advance is used to implement CEM’s quick recovery mechanism, but a number of other applications could almost certainly be devised. An important final note is that these experiments are not provably robust, due to time constraints and the significant amount of time taken to simulate a wireless sensor network in detail. All information presented above should be reliably verified before being used.

8 Future Work

Future work could be taken in a number of directions. Obvious improvements to the CEMS protocol are complete implementation and configuration of a k-means algorithm instead of the existing genetic algorithm. A method for rotating cluster heads during the steady-state phase without interrupting data reporting would be a useful feature for scenarios that have widely varying transmission costs between clusters. Support for dynamically determining the reclustering period would not only reduce configuration, but better distribute energy load as sensors die and cluster sizes change. Reference broadcasts during slot 0 could be implemented to deal with clock drift. With the inclusion of retransmission rules (which is currently assumed to be handled by the data-link layer), CEMS could nearly supplant many MAC protocols and operate on its own. Finally, an algorithm to analytically determine the optimal number of clusters given the current network topology would remove the need for almost any configuration, as well as better adapting the cluster-selection mechanism to changing environments and sensor populations.

Enhancements to the simulation environment might reveal previously unexplored avenues of improvement, as well. More advanced radio models capable of simulating multi-path fading, and a more realistic portrayal of the wireless environment (e.g. channel noise), could be invaluable in tuning the global TDMA schedule. Simulation of an underlying environment to be sensed could allow CEMS to be investigated in the context of application-specific aggregation and compression algorithms. Inclusion of energy harvesters such as solar panels would allow a variety of currently excluded network configurations to be investigated.

Future experiments, in addition to those described above, might investigate the relationship between data packets and control overhead. Smaller data packets representing simple, floating-point sensor measurements might result in control packet transmission and reception dissipating significant amounts of energy. Quantification of this relationship might allow for inefficiencies to be better identified and addressed. Investigating the ability of CEMS to handle sensors with heterogeneous and fluctuating residual energy (due to, e.g. energy harvesters or mains-powered gateway nodes) could reveal more efficient cluster-selection techniques which consider more than simple spatial proximity. Finally, only 100 sensor nodes were ever simulated at once. An investigation of CEM’s behavior in more populated networks (e.g. 250 or even 1000 nodes) would be a valuable insight into the protocol’s scalability, as well as its ability to perform in dense network topologies.

9 References

[1] Heinzelman, W.; Chandrakasan, A.; Balakrishnan, H., "Energy-efficient communication protocol for wireless microsensor networks," System Sciences, 2000. *Proceedings of the 33rd Annual Hawaii International Conference*, 10 pp. vol.2-, 4-7 Jan. 2000

[2] Heinzelman, W., Chandrakasan A., Balakrishnan H. "An Application-Specific Protocol Architecture for Wireless Microsensor Networks." *IEEE Transactions on Wireless Communications* 1 (2002).

[3] Voigt, T.; Dunkels, A.; Alonso, J.; Ritter, H.; and Schiller, J. 2004. “Solar-aware clustering in wireless sensor networks”. In *Proceedings of the Ninth international Symposium on Computers and Communications 2004 Volume 2 (Iscc"04) - Volume 02* (June 28 - July 01, 2004). ISCC. IEEE Computer Society, Washington, DC, 238-243.

[4] Manjeshwar, A. and Agrawal, D.P. “Teen: a routing protocol for enhanced efficiency in wireless sensor networks”. In *Parallel and Distributed Processing Symposium., Proceedings 15th International*, pages 2009-2015.

[5] Tang, Q.; Tummala N.; Gupta S.,;and Schweibert L. “Communication Scheduling to Minimize Thermal Effects of Implanted Biosensor Networks in Homogeneous Tissue”. *IEEE Transcations on Biomedical Engineering* 52 (2005): 1285-1293.

[6] Mudundi, S.R., and Hasham H.A. “A New Robust Genetic Algorithm for Dynamic Cluster Formation in Wireless Sensor Networks”. *Proceedings of the Seventh IASTED International Conferences* (2007).

[7] Hussain S.; Matin A.W.; Islam O., "Genetic Algorithm for Energy Efficient Clusters in Wireless Sensor Networks”," pp.147-154, *International Conference on Information Technology* (ITNG'07), 2007

[8] Grefenstette, J., "Optimization of control parameters for genetic algorithms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-16(1), pp. 122-128, 1986.

[9] Haupt, R. "Optimum population size and mutation rate for a simple real genetic algorithm that optimizes array factors." *Proc. of Antennas and Propagation Society International Symposium*, 2000, Utah, Salt Lake City.

[10] Varga, A. *Omnet++ Discrete Event Simulation System*. Computer software. Vers. 3.2. Omnet++ Community Site. <http://www.omnetpp.org/>.

[11] Mobility Framework for Omnet++. Computer software. Vers. 2.0p3. *Mobility Framework*. <http://mobility-fw.sourceforge.net/>.

[12] Chong, C. and Kumar, S. "Sensor Networks: Evolution, Opportunities, and Challenges." *Proceedings of the IEEE* 91 (2003).

[13] "The Cooperative Engagement Capability." *John Hopkins APL Technical Digest* 16 (1995).

[14] DN2510. 2007. Dust Networks. <http://www.dustnetworks.com/cms/sites/default/files/DN2510.pdf>.

[15] Imote 2. Crossbow Technologies. <http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/Imote2_Datasheet.pdf>.

[16] Murty R.; Mainland G.; Rose I.; Chowdhury A.; Gosain A.; Bers J.; and Welsh M.. "CitySense: A Vision for an Urban-Scale Wireless Networking Testbed." *Proceedings of the 2008 IEEE International Conference on Technologies for Homeland Security* (2008).

[17] TC74. Microchip. <http://ww1.microchip.com/downloads/en/devicedoc/21462c.pdf>.

[18] Omnivision OV7640. Datasheet Archive. <http://www.datasheetarchive.com/OV7640-datasheet.html>.

[19] MSP430. Texas Instruments. <http://www.snm.ethz.ch/pub/uploads/Projects/MSP430_datasheet.pdf>.

[20] AVR. Atmel. <http://www.snm.ethz.ch/pub/uploads/Projects/atmel_atmega128l_datasheet.pdf>

[21] XScale. Intel. <http://download.intel.com/design/intelxscale/XScaleDatasheet4.pdf>.

[22] CC1021. Texas Instruments. <http://focus.ti.com/lit/ds/symlink/cc1021.pdf>.

[23] CC1000. Texas Instruments. <http://focus.ti.com/lit/ds/symlink/cc1000.pdf>.

[24] CC2500. Texas Instruments. <http://focus.ti.com/lit/ds/symlink/cc2500.pdf>.

[25] Levis P.; Madden S.; Polastre J.; Szewczyk R.; Whitehouse K.; Woo A.; Gay D.; Hill J.; Welsh M.; Brewer E.; and Culler D.; "Tinyos: An operating system for sensor networks," 2005, pp. 115-148.

[26] Zeng X.; Bagrodia R.; Gerla M.; "GloMoSim: a Library for Parallel Simulation of Large-scale Wireless Networks". *Proceedings of the 12th Workshop on Parallel and Distributed Simulations -- PADS '98*, May 26-29, 1998 in Banff, Alberta, Canada

[27] Mccanne, S; Floyd, S.; and Fall, K. ns2 (network simulator 2). http://www-nrg.ee.lbl.gov/ns/.

[28] Varga A. and Hornig R., "An Overview of the Omnet++ Simulation Environment," in *Simutools '08: Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*. ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008, pp. 1-10.

[29] Levis, P.; Lee N.; Welsh. M.; and Culler, D., "Tossim: Accurate and Scalable Simulation of Entire TinyOSs Applications," in *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*. New York, NY, USA: ACM Press, 2003, pp. 126-137.

[30] Lin, K.; Hsu, J.; Zahedi, S.; Lee, D.C.; Friedman, J; Kansal, A.; Raghunathan, V.; Srivastava, M.B., "Heliomote: Enabling Long-Lived Sensor Networks Through Solar Energy Harvesting," *ACM Sensys* , November 2005.

[31] Chulsung, P. and Chou, P.H. "AmbiMax: Autonomous Energy Harvesting Platform for Multi-Supply Wireless Sensor Nodes." *Proc. of Sensor and Ad Hoc Communications and Networks*, 2006., Virginia, Reston.

[32] Pahlavan, K. and Krishnamurthy, P., “Principles of Wireless Networks – A Unified Approach”, 2ed Edition, John Wiley and Sons, 2008.

[33] Halkes, G. and Langendoen, K. "Crankshaft: An Energy-Efficient MAC-Protocol For Dense Wireless Sensor Networks." *Proc. of 4th European conference on Wireless Sensor Networks,* Netherlands, Delft.

[34] Mount, David. “KMlocal: A Testbed for k-means Clustering Algorithms”. 10 Aug. 2005. University of Maryland. <http://www.cs.umd.edu/~mount/Projects/KMeans/kmlocal-doc.pdf>.

10 Appendix I: Parameter Settings

Our simulations relied on a variety of parameters to simulate the environment, wireless medium, CEMS protocol, and radio electronics. All values specific to the physical layer are based on the CC2500 transceiver module:

Physical and Data Link Layer

- Signal attenuation threshold: -100dBm
- Sensor transmitter power: 1mW
- Base station transmitter power: 100mW
- E_{elec}: 50nj/bit
- ε_{amp}: 100pJ/bit/m2
- k: 4000 bits
- bit rate: 250kbps
- MAC header: 34 bytes

CEMS and Network Layer

- Data packets: 500 bytes
- Layer-3 header: 24 bytes
- TDMA cycle time: 10s
- TMDA slot length: 50ms
- Missed transmission count limit : 2
- Number of clusters: 10% of the original population
- Reclustering Period: 20 hours
- Poisson death rate: 5 expected deaths per day

Genetic Algorithm

- Number of generations: 5000
- Mutation rate: 0.0005
- Crossover rate: 0.06
- Population Size: 100