

# ALARM-NET: Wireless Sensor Networks for Assisted-Living and Residential Monitoring

A. Wood, G. Virone, T. Doan, Q. Cao, L. Selavo, Y. Wu, L. Fang, Z. He, S. Lin, J. Stankovic  
Department of Computer Science  
University of Virginia

{wood|gv6f|tnd7c|qc9b|selavo|yw5s|leifang|zh5f|sl8yc|stankovic}@cs.virginia.edu

## Abstract

We describe ALARM-NET, a wireless sensor network for assisted-living and residential monitoring. It integrates environmental and physiological sensors in a scalable, heterogeneous architecture. A query protocol allows real-time collection and processing of sensor data by user interfaces and back-end analysis programs. One such program determines circadian activity rhythms of residents, feeding activity information back into the sensor network to aid context-aware power management, dynamic privacy policies, and data association. Communication is secured end-to-end to protect sensitive medical and operational information.

The ALARM-NET system has been implemented as a network of MICAz sensors, stargate gateways, iPAQ PDAs, and PCs. Customized infrared motion and dust sensors, and integrated temperature, light, pulse, and blood oxygenation sensors are present. Software components include: TinyOS query processor and security modules for motes; AlarmGate, an embedded Java application for managing power, privacy, security, queries, and client connections; Java resident monitoring and sensor data querying applications for PDAs and PCs; and a circadian activity rhythm analysis program.

We show the correctness, robustness, and extensibility of the system architecture through a scenario-based evaluation of the integrated ALARM-NET system, as well as performance data for individual software components.

## 1 Introduction

An aging baby-boom generation is stressing the U. S. healthcare system, causing hospitals and other medical caregivers to look for ways to reduce costs while maintaining quality of care. It is economically and socially advantageous to reduce the burden of disease treatment on the system by enhancing prevention and early detection. This requires a long-term shift from a centralized, expert-driven, crisis-care model to one that permeates personal living spaces and in-

volves informal caregivers, such as family, friends, and community.

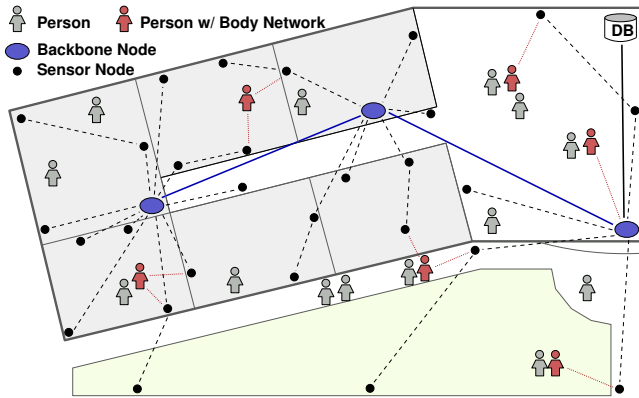
Systems for enhancing medical diagnosis and information technology often focus on the clinical environment, and depend on the extensive infrastructure present in traditional healthcare settings. The expense of high-fidelity sensors limits the number available for outpatient deployment, and some require specialized training to operate. Manual record keeping has been identified as a key source of medical errors [1], and at its best, traditional data collection is intermittent, leaving gaps in the medical record.

Wireless Sensor Networks (WSNs) can provide capabilities that are valuable for continuous, remote monitoring, as the research into military [2] and environmental [3] systems attest. For healthcare applications, they can be deployed inexpensively in existing structures without IT infrastructure. Data is collected automatically, enabling daily care and longitudinal medical monitoring and diagnosis. The wireless devices can integrate with a wide variety of environmental and medical sensors.

While addressing some of the needs of distributed healthcare, they also present their own challenges to being practical, robust platforms for pervasive deployment. In multi-resident dwellings, associating collected data with the correct person is difficult and inexact. Privacy and security of medical data collected may be jeopardized by use of a wireless medium. Without smart power management, battery-powered sensors have short lifetimes of a few days or require continual maintenance.

We present ALARM-NET, an Assisted-Living and Residential Monitoring Network for pervasive, adaptive healthcare. Figure 1 shows an example deployment of the system in an assisted-living community with many residents or patients. Contributions of the work include:

- An extensible, heterogeneous network architecture that addresses the challenges of an ad hoc wide-scale deployment, and integrates embedded devices, back-end systems, and user interfaces,
- Context-aware protocols informed by Circadian activity rhythm analysis, which enable smart power management and dynamic alert-driven privacy tailored to the individual's patterns of activity,
- Query protocol for streaming online sensor data to user interfaces, integrated with privacy, security, and power management.



**Figure 1. Assisted-living deployment example, showing connections among sensors, body networks, and backbone nodes.**

- SecureComm, a hardware-accelerated secure messaging protocol and TinyOS module that supports user-selectable security modes and multiple keys,
- A system implementation and evaluation using custom and commodity sensors, embedded gateway, and back-end database and analysis programs.

After reviewing related work and describing the ALARM-NET architecture, each major component of the system is separately detailed: query management, Circadian activity rhythms, dynamic privacy, data and system security, context-aware power management, and data association in Sections 4–9. Implementation details are in Section 10, followed by an evaluation of the system’s performance and a conclusion.

## 2 Related Work

In the presence of increasing numbers of aging populations there is a significant interest in smart environments and living spaces that monitor and assist individuals. Such systems monitor vital signs as well as attempt to learn the context of the events happening during the lives of the inhabitants of the environment. Several such systems are described in this section.

Researchers at Intel Research Seattle and the University of Washington have built a prototype system that can infer a person’s activities of daily living (ADLs) [4]. Sensor tags are placed on everyday objects such as a toothbrush or coffee cup. The system tracks the movement of tagged objects with tag readers. The long-range goal is to develop a computerized and unobtrusive system that helps with managing ADLs for the senior population [5].

University of Rochester is building The Smart Medical Home [6], which is a five-room “house” outfitted with infrared sensors, computers, bio-sensors, and video cameras for use by research teams to work with research subjects as they test concepts and prototype products. Researchers observe and interact with subjects from two discreet observation rooms integrated into the home. The goal is to develop an integrated Personal Health System that collects data 24 hours a day and presents it to the health professionals.

Georgia Tech built an Aware Home [7] as a prototype

for an intelligent space. This space provides a living laboratory that is capable of knowing information about itself and the different types of activities of its inhabitants. It combines context-aware and ubiquitous sensing, computer vision-based monitoring, and acoustic tracking together for ubiquitous computing for everyday activities while remaining transparent to the users.

Massachusetts Institute of Technology (MIT) and TIAX, LLC are working on the PlaceLab initiative [8], which is a part of the House\_n project. The mission of House\_n is to conduct research by designing and building real living environments—“living labs”—that are used to study technology and design strategies in context. The PlaceLab is a one-bedroom condominium with hundreds of sensors installed in nearly every part of the home.

Researchers at Harvard have developed a suite of wireless sensors and software called CodeBlue for a range of medical applications, including pre-hospital and in-hospital emergency care, disaster response, and stroke patient rehabilitation. The sensors include portable 2-lead ECG, pulse oximeter, wearable Pluto mote with builtin accelerometer, and a module with accelerometer, gyroscope, and electromyogram (EMG) sensor for stroke patient monitoring. In addition to the hardware platform, a scalable software infrastructure for wireless medical devices is designed to provide routing, naming, discovery, and security for wireless medical sensors [9].

University of Washington’s Assisted Cognition project incorporates novel computer systems enhancing the quality of life of people suffering from Alzheimer’s Disease and similar cognitive disorders. This project combines computer science research in artificial intelligence and ubiquitous computing with clinical research on patient care. Assisted Cognition systems are proactive memory and problem solving aids that help an individual perform daily tasks by sensing the individual’s location and environment, learning to recognize patterns of behavior, offering audible and physical help, and alerting caregivers in case of danger [10].

Similarly to the systems described above, ALARM-NET monitors environmental and physiological data of individuals in their residences, with focus on the assisted-living and medical domains. Unlike other systems, ALARM-NET incorporates a Circadian Activity Rhythm (CAR) analysis module that learns the patterns of daily life of the individuals, and influences the system and network protocols for power management and privacy. For example, the dynamic privacy configuration rules change on the fly when an individual exhibits a behavior that is critical to his health and enable the authorized medical personnel to access vital data, which is otherwise hidden or available for anonymous statistical purposes only.

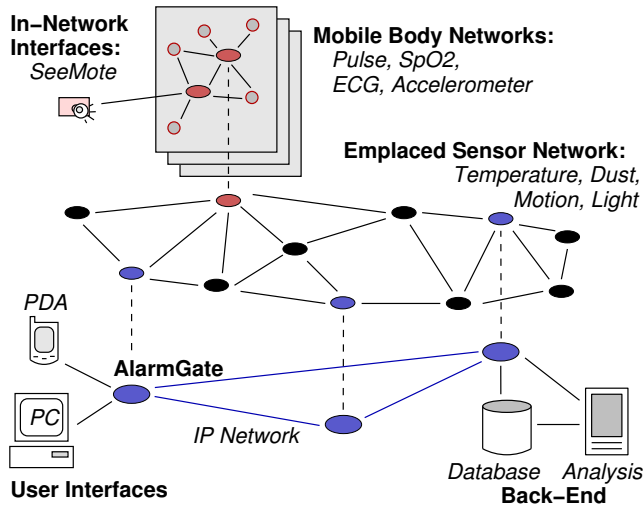
Additionally, CAR enables advanced power management by anticipating which sensors should be kept active and which can be temporarily disabled in order to conserve power according to the habits of the individual.

Another constructive aspect of the system is a flexibility that allows easy integration of new sensors or even mobile networks of sensors. ALARM-NET is a multi-platform, security- and privacy-aware architecture integrating a wide

spectrum of heterogeneous devices such as wireless sensor nodes, gateways, PDAs, and back-end systems for data storage, data association, and CAR analysis.

### 3 ALARM-NET Architecture

ALARM-NET integrates heterogeneous devices in a common architecture, spanning wearable body networks, emplaced wireless sensors, and IP-network elements. The high-level architecture can be described by categorizing devices based on their platform and role in the system. Each category is shown in Figure 2 and described below.



**Figure 2. ALARM-NET architecture components and logical topology.**

**Body Networks** are wireless sensor devices worn by a resident which provide physiological sensing or activity classification. Each patient can wear a body network tailored to his own medical needs. In addition, graphical or audible notifications to the patient (for example, alerts to take medicine) are made using an in-network wearable interface, the SeeMote we developed [11] with a color LCD.

We have also integrated SATIRE [12], a body network that classifies Activities of Daily Living (ADLs) [4] by analyzing accelerometer data generated by movements of the wearer.

Data from the body network is transmitted through the emplaced sensors to user interfaces or back-end programs.

**Emplaced Sensors** include devices deployed in the living space to sense environmental quality or conditions, such as temperature, dust, motion, and light. Motion, in particular, provides a spatial context for patient activities, enabling location tracking and data association.

These devices form a multi-hop wireless network to the nearest AlarmGate application running on a stargate. Depending on the deployment environment, they may use wired or battery power or a combination. Sensor nodes answer queries for local data and perform limited processing and caching.

Emplaced sensors maintain connections with mobile body networks as they move through the living space, so that queries and data reports are not interrupted.

**AlarmGate** applications run on embedded platforms, managing system operations and serving as application-level gateways between the wireless sensor and IP networks. These nodes allow user interfaces to connect, authenticate, and interact with the system.

Software modules for dynamic privacy, power management, query management and security reside on the AlarmGate. A connection to a back-end database provides long-term storage of data and configuration. Other back-end analysis programs connect as clients to update context in the system.

**Back-end** systems provide online analysis of sensor data and long-term storage of system configuration, user information, privacy policies, and audit records.

A Circadian Activity Rhythm (CAR) analysis program processes sensor data stored in the database, learning individual behavior patterns. These are fed back into the network to aid context-aware power management and privacy.

**User Interfaces** allow any legitimate user of the system (doctors, nurses, patients, family, etc) to query sensor data, subject to enforced privacy policies. We developed a patient-tracking program suitable for a nurses station, and a query issuer that runs on a PDA and graphs real-time sensor data.

Figure 1 shows an example of a deployment in an assisted-living environment. AlarmGate applications connect via the IP network to each other and the central database. Emplaced sensors and body networks form an ad hoc multi-hop network throughout the assisted-living community.

To better illustrate how this architecture supports assisted-living and longitudinal studies, consider the following scenario. A husband and wife live together in an assisted-living facility. The husband has heart problems and wears a wireless ECG. His wife has bouts of falling and wears accelerometers embedded in clothes.

Sensor data for both are periodically collected and transmitted to nurse stations and back-end databases. A doctor is in another part of the assisted-living facility and queries for heart information on the husband from a PDA. The doctor is authenticated and privacy rights are checked. If everything is in order, the emplaced sensor network and AlarmGates will act as a communication network and enable the doctor to see a real-time display of the ECG on his PDA.

In the meantime, the wife falls. This is detected immediately and an alarm is sent to the nurse's station. Data, possibly filtered and aggregated, is collected in the back-end database where not only raw patient information is collected, but also the context under which this data was collected. This context includes the medical history of the patient, the recent activities of the patient and the current environmental conditions. Such data can support long term medical studies.

Connected to the back-end database, a circadian rhythm program analyzes typical behaviors for the husband and wife. If irregularities in their daily living occur, an alert is issued to investigate whether some medical problem is causing the irregularity. For example, perhaps the husband is using the bathroom twice as often as normal.

## 4 Query Management

Real-time data queries are an important functionality in ALARM-NET, enabling user interaction with the running system and automatic data collection. In both cases, a flexible query protocol is used. Sensor devices may service multiple ongoing queries from the system and users simultaneously.

Queries are identified by  $\langle \text{source, ID} \rangle$  tuples, and request a certain type of sensor data about a subject. If the subject is a user, it is translated to a particular sensor or the AlarmGate, by consulting static sensor configuration or the current location of the subject. Access to a subject's sensor data is subject to authorization by the Privacy Manager, and depends on the configured policies and current context of the subject.

For a single-shot query, the sensor samples the requested data and returns a single report to the originator, completing the transaction.

Periodic queries are issued with a given sample period. As data is collected, reports are streamed back to the requester until a stop command is received or an optional duration value is reached. Later the query may efficiently be restarted with a reissue command.

A separately specified report period allows multiple samples to be collected and aggregated into a single report using various lightweight aggregation functions. They may also be filtered by a user-provided filter function and value.

Using lightweight aggregation and filter functions at the source of the generated data reduces report frequency, saving energy. Further efficiency is gained by allowing as many samples as will fit to be combined into a single report message. This caching saves messaging overhead and reduces radio traffic.

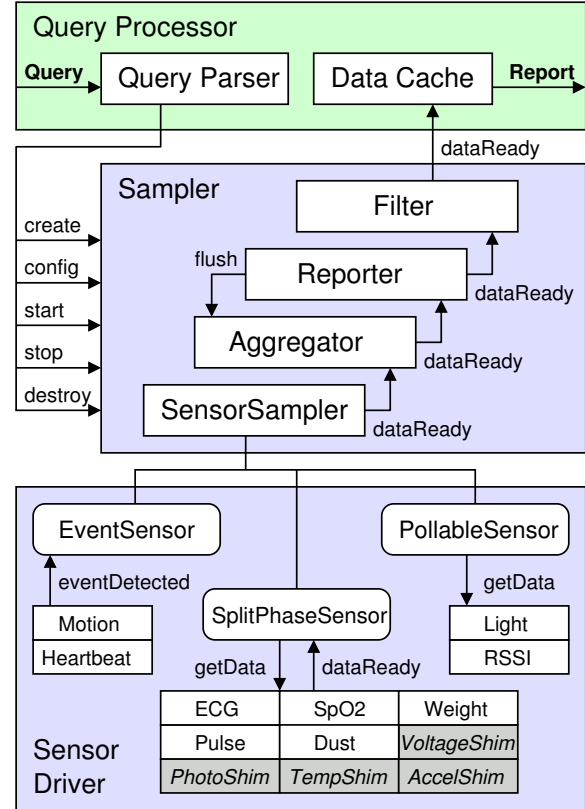
In addition to the generated data, reports bear a timestamp. Reports with multiple samples include the timestamp of the last report. Reports may signal success with an ACK, or errors with a NACK and cause code.

Each sensor type may generate individual sample readings of different lengths. The default data-size may be overridden by the user since domain knowledge may allow greater packing efficiency, and because some of the aggregation functions (like `sum` and `count`) may overflow the default data-size of their sensor types.

For integration with power management, queries are given a priority. Background, non-critical system queries have low priority, while user requests have high priority. Queries with lower priority than power management decisions may be denied.

**Software Components** run on the AlarmGate and on nodes. At the AlarmGate, the Query Manager interfaces with clients that originate queries. The properly translated and authorized query is sent to the sensor(s) in the WSN, where it is parsed by a Query Processor, shown in Figure 3.

Sampler is a combination of modules comprising a data generation and filter chain. The SensorSampler module maintains a schedule for sensor sampling to satisfy multiple ongoing queries. It interfaces with sensors using three types of interfaces: EventSensor, SplitPhaseSensor, and PollableSensor. All sensor types in ALARM-NET belong to one of



**Figure 3. Query processing stack on sensor devices. The Query Processor parses queries, and starts the Sampler, which reads data from the sensor drivers on schedule, generating data that flows up the processing chain toward the Query Processor for reporting.**

these categories. Third-party sensor drivers are wrapped in a shim which conforms to this interface.

As each sample is available (asynchronously, after ADC conversion, or immediately for the interfaces, respectively), it propagates up a processing chain. The user-specified aggregation function operates on every sample, and is reset or flushed when a report is generated. This allows the query to specify the number of samples to be aggregated into each report.

Sampler-reports are further subject to a filter function. If the filter predicate is satisfied, the report continues up to the Query Processor. Here it is cached (if requested) or timestamped and sent back across the WSN to the originator.

For example, pulse-rate samples may be collected every  $200\text{ms}$  but reported only every  $1\text{s}$ , each an average of five samples. These may further be filtered by discarding them unless the pulse-rate is above  $130\text{bpm}$ .

**Queries** are originated by the system or by users, and contain the elements shown in Figure 4. Sensors generate reports according to the given sample and report periods and processing functions.

Relational filter functions require the user to include a four-byte value for comparison. This field is divided into a high and low portion for the range comparison.

Default units for sample and report periods are millisec-

QUERY	=	<source, id, <i>command</i> >
<i>command</i>	=	{ <i>periodic</i>   <i>single</i>   stop   reissue }
<i>periodic</i>	=	<sensorType, priority, <i>aggregate</i> , <i>filter</i> , dataSize, cacheFlag, unitsSeconds, samplePeriod(3), reportPeriod(3), [filterValue(4)], [duration] >
<i>single</i>	=	<sensorType, priority, dataSize>
<i>aggregate</i>	=	{ none   count   sum   diff   max   min   mean   latch }
<i>filter</i>	=	{ none   $x < \text{value}$   $x > \text{value}$   valueHi $> x >$ valueLo   changed }
REPORT	=	<id, <i>type</i> >
<i>type</i>	=	{ <i>data</i>   ack   nack }
<i>data</i>	=	<timestamp(4), data(1-4), ... >

**Figure 4. Query and Report message contents. Lengths of multi-byte fields are given.**

onds, but can be interpreted as seconds for long-running queries by setting the `unitsSeconds` flag.

Single, stop, and reissue commands are designed to be as small as possible, for maximum efficiency.

## 5 Circadian Activity Rhythms

It is known that most people exhibit a behavioral trend within the home, called a ‘‘Circadian Activity Rhythm’’ (CAR). We have developed a CAR analysis program that measures the rhythmic behavioral activity of patients and detects any behavioral changes within these patterns. We employ CAR in novel ways for both improved medical care and for improved network performance. In particular, CAR supports context-aware protocols based on these activities for dynamic alarm-driven privacy and smart heterogeneous power management (Sections 6,8).

The CAR algorithm is statistical and predictive. First presented in [13], it is based on the distribution of the probability of user presence in every room. CAR runs on the back-end of the system on a PC, and polls a database where patient activity is stored.

CAR supports a GUI, which displays various information related to the activity analysis, such as the number of abnormal time periods (under-presence or over-presence in a room) that occurred per hour and day during day or night, and the length and dates of the stay of the resident.

Other graphs of the GUI display the main results of the CAR analysis. The graphs in 5 present a real clinical case study for a healthy resident who stayed 25 days in an assisted-living facility. The first one, on the left, displays the average time the user spends in every room each hour, calculated over the number of days of the stay of the resident. The second graph, on the right, indicates that after 18 days the circadian rhythm for this patient has been learned. These graphs can provide a wealth of information about activity patterns such as the sleep/wake cycle, or some medical hints to the physician about some activities of daily living (ADLs) [4] of the resident such as eating, hygiene and sleeping. In the future, more specific ADLs will also be collected.

To use CAR for health monitoring, after the learning pe-

riod, any statistically significant anomalies that are detected will alert physicians who can investigate the source of the trouble (sleep/wake period longer, one meal less, etc.) by focusing on the region of the anomaly as identified by CAR. The hypothesis is that behavioral changes could, in the long-term, reveal health decline or pathologies. This hypothesis was clinically validated in collaboration with the MARC Center at the University of Virginia medical school. To validate this hypothesis, clinical behavioral patterns of older adults in assisted-living facilities were extracted from real data sets, and behavioral changes were studied by consulting the medical notebooks of the caregivers in charge of the monitored residents. See [13] for details of this validation.

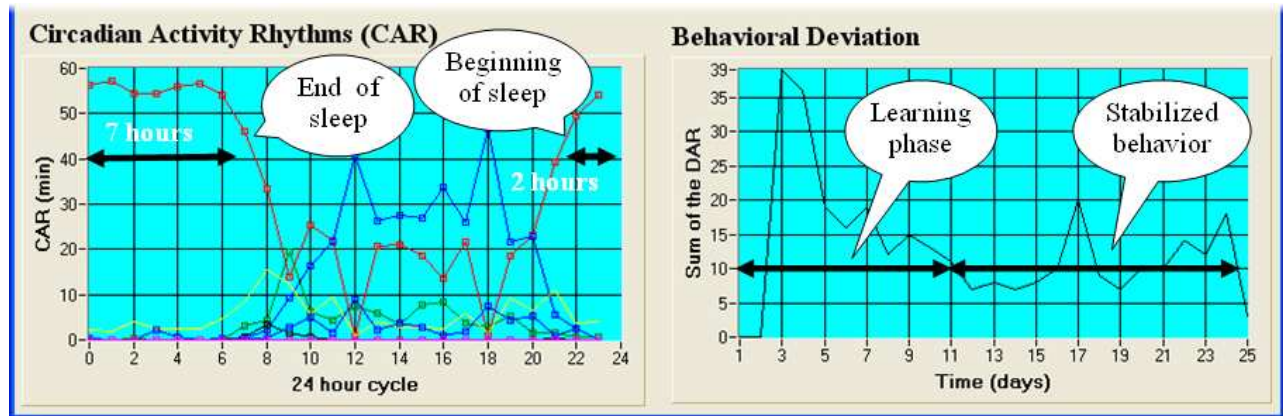
## 6 Dynamic Context-Aware Privacy

Data collected by assisted-living and residential monitoring applications is abundant and frequent. It can reveal intimate details about a person’s living activities and health status. As wireless sensor networks grow stronger in their capability to collect, process, and store data, personal information privacy becomes a rising concern. Our system includes a framework to protect privacy and still support the need to provide timely assistance to patients in critical health conditions.

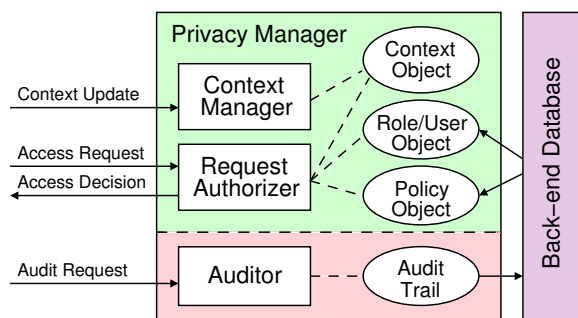
Emergency-aware applications demand a privacy protection framework capable of responding adaptively to each patient’s health conditions and privacy requirements in real-time. Therefore, traditional role-based access control which makes access authorization based on users’ static roles and policies is not flexible enough to meet this demand. We implement a privacy protection framework which is dynamically adjustable to users’ context, allows data access authorization to be evaluated on the fly, and is able to adapt to patients’ emergency cases.

A key novelty is that access rulings can be dynamically altered based on contexts generated by the CAR algorithm when necessary. For example, if a patient has blocked access to their ECG data for nurses, but the CAR has determined serious anomalous behavior that might indicate a heart problem, then the nurse is alerted and access to the data is allowed for a period of time. In case of an alarming health status, alarms are sent over the network to restrict or relax the privacy depending on the user’s role. Other contexts in our system include the patient’s physiological conditions (ECG and pulse readings), living environment conditions (room temperature and light), and autonomy (inferred from ADLs by the CAR). Also, our system is extensible to incorporate contexts from different analysis algorithms at both the back-end server and the front-end network of wireless devices. System designers can specify the privacy policies at different levels of granularity from individual sensors to individuals or groups of users.

The main component of the privacy framework is the Privacy Manager module which receives data access requests from the Query Manager module and makes authorization decisions based on real-time context. Context, identified by the tuple <context id, context subject, context value>, is the result of domain expert analysis based on sensor readings which indicate a patient’s health and environment conditions.



**Figure 5. Circadian Activity Rhythms analysis GUI.** Average time spent in every room per hour is graphed on the left side. Sums of daily deviations from the user’s norm are on the right, showing a learning period after initial deployment.



**Figure 6. Privacy-related components in ALARM-NET.**

The Privacy Manager resides in the AlarmGate application and has three main functional components: the Context Manager, the Request Authorizer, and the Auditor (see Figure 6).

- **Context Manager:** collects and maintains the context objects about users and the environment from different analysis modules in the system.
- **Request Authorizer:** queries received at the Query Manager are forwarded to the Request Authorizer which makes access decisions by consulting the system’s privacy policies and context objects of the query subject. After each access request is decided at the Request Authorizer, it is recorded by the Auditor module.
- **Auditor:** maintains a trace of access requests in an audit trail, including the authorization decision made for each request (granted or denied).

## 7 Network and Data Security

Security of medical records and data is a vital part of ALARM-NET. By design, the system allows the collection and storage of environmental, physiological, behavioral, and location data about residents. This data must be protected against unauthorized disclosure, especially given that much of the system uses wireless communication, which is vulnerable to compromise from exterior spaces or adjacent living units.

Access to an AlarmGate by user interfaces is limited to legitimate users of the system, who must authenticate themselves before being allowed to continue. Queries for sensor

data are authorized according to administratively configured privacy policies and context, described in Section 6. Authorization considers only whether a user may perform some action, hence authentication, determining whether a client is who it claims to be, must precede it.

After a client connects and authenticates, communication between it and the AlarmGate on the IP network is encrypted whenever sensor data is reported. Messages sent and received to/from the WSN by the AlarmGate must also be secured, using message authentication codes (MACs) and encryption when necessary.

Finally, the connection to the back-end database must also be resistant to attacks, a common requirement and one that is addressed by commercially available programs. We therefore do not discuss it further.

Connections to traditional systems on IP networks must use authentication and encryption methods adequate for protection against attackers typically seen there. This requires relatively heavyweight methods compared to those feasible on the much more restricted WSN platform. Here we use lightweight protocols with hardware accelerated cryptography to reduce power consumption and overhead on already constrained devices.

We discuss security mechanisms in ALARM-NET for communication with IP clients, communication with and among WSN devices, and processing inside the AlarmGate.

**IP Network Security** involves connections to the AlarmGate from potentially any Internet host. We therefore must not trust any messages received until the remote party has been properly authenticated.

We use the freely available Secure Remote Password (SRP) Protocol to authenticate clients to the AlarmGate server [14]. It provides several beneficial properties:

- The server does not directly store the password, so in the event of a database or AlarmGate compromise, the attacker cannot use data to directly compromise a host.
- As a by-product of authentication, the server and client have shared a session key, which is used to provide a secure channel between them. Since the session key is randomly generated, SRP provides perfect forward secrecy.

- Values exchanged during authentication cannot be used for impersonation or to compromise the session key by an eavesdropping party.
- SRP does not require a trusted third party, like a certification authority.

The server and client agree *a priori* on a modulus and generator, according to [14]. The back-end database stores the tuple <username, verifier, salt>, where the salt is generated randomly when the user is enrolled, and the verifier is derived from his password.

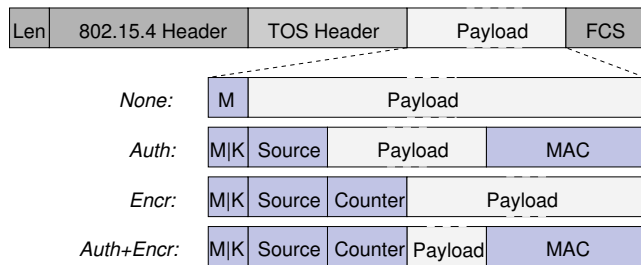
After successful authentication, the session key is used with AES (Advanced Encryption Standard) [15] modes for authentication and/or encryption. These were chosen to be compatible with the SecureComm security suite described below.

**WSN Security** is provided by SecureComm, a link-layer security suite we developed for MICAz and Telos motes. These both use the Chipcon CC2420 radio transceiver, which has built-in support for inline cryptographic operations conforming to the 802.15.4 standard [16].

SecureComm uses this hardware-accelerated cryptography in a backward-compatible manner with insecure TinyOS messages. It supports multiple keys per source, allowing it to operate with the many key management schemes found in current literature.

AES Security modes supported by the protocol are those with hardware support: none, CBC-MAC authentication-only [17], CTR mode encryption-only [18], and CCM [19], which combines authentication with encryption.

These modes are selectable on a per-message basis, allowing application-level decisions about security policy based on message semantics or context, which is more flexible than other solutions ([20, 21]).

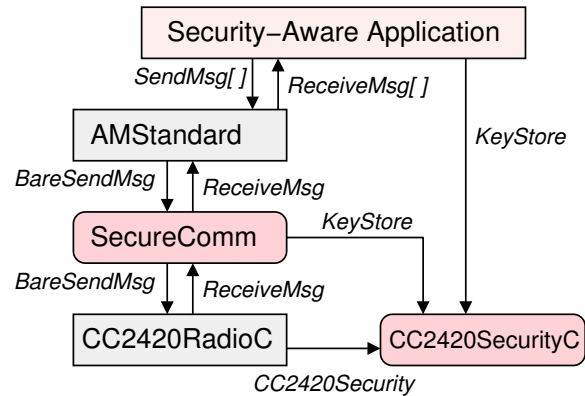


**Figure 7. Message format for the SecureComm modes.** *M|K* is a bitfield consisting of the mode and key id.

Each SecureComm message contains only the header fields necessary for the desired mode of operation. Message formats are shown in Figure 7. At a minimum, an additional byte specifying a mode of NONE is necessary. For authentication-only mode, a key id and two-byte source are added, as well as the four-byte message authentication code (MAC). Encryption or encryption-and-authentication modes also include a two-byte non-repeating counter.

Keys are identified in the AlarmGate and sensor devices by <source, key-id> tuples, stored in the KeyStore module. The KeyStore also manages monotonically increasing transmit and receive counters, which is important to preserve the security of CTR and CCM modes.

SecureComm is a link-layer security suite, and the 802.15.4 and TOS headers are included in MAC computations. Therefore, neighbors share keys with each other for authentication and/or encryption for messages shared among them. Messages from an attacker (that claim to be secure) are discarded when received by a legitimate node in the network, since with high probability the MAC will not be computed correctly.



**Figure 8. SecureComm component wiring.**

Figure 8 shows how the TinyOS components are wired together for an application that uses SecureComm. Prior to sending or receiving a secure message, the application must store a key in the CC2420SecurityC module using the *KeyStore* interface. When sending a message, the application specifies the security mode and key-id in the message structure. The SecureComm component is interposed between AMStandard and the radio configuration, where it adds and removes the secure message headers/footers shown in Figure 7 as needed. When receiving or transmitting, CC2420RadioC consults CC2420SecurityC using the *CC2420Security* interface to setup parameters for inline security operations.

Backward compatibility is preserved by segmenting the TinyOS message type field. A compile-time constant determines the threshold separating standard Active Messages from SecureComm messages. SecureComm processing is only invoked when receiving messages above the specified threshold. This allows components unaware of security to be re-used in the network unchanged (after verifying their message types are below the threshold), albeit without security guarantees.

Configuration, queries, and other important commands from AlarmGate into the WSN are authenticated. Reports are authenticated and encrypted to preserve data confidentiality. Various non-critical messages may use the insecure mode.

**Internal Security** functions in the AlarmGate application include key management and auditing.

Keys for each WSN node and each connected client are managed by a KeyStore. Each is identified by <source, key-id>, as described above. AlarmGate shares a key with each sensor device that is a direct neighbor.

Incoming secure messages from either network require a

lookup to find the appropriate key in the KeyStore, followed by decryption and/or MAC verification according to the security mode selected by the sender.

Outgoing messages also require a key lookup, followed by MAC computation and/or encryption if indicated. Maintenance of transmit counters is automatic, but receive counters must be manually updated since message loss causes discontinuities.

Message and user authentication failures are stored in the Auditor’s Log, shared with the Privacy Manager, for review by administrators to aid in diagnosing ongoing attacks.

## 8 Context-Aware Power Management

ALARM-NET supports a heterogeneous power situation where some nodes are plugged into the wall and others operate on batteries. Stargates are plugged-in due to their high power consumption requirements, while motes are either plugged-in or operate on batteries. This flexibility allows new battery-operated motes to be added quickly, in high density if needed, moved easily, and even outdoors where power is more problematic. Further, motes that are part of body networks must necessarily work on batteries (or scavenge energy from motion). Consequently, energy efficiency is an important design issue in ALARM-NET.

Application demands of ALARM-NET pose some particular requirements on power management. First, power management should be adaptive to the resident’s behavior. Sensors in ALARM-NET are used to detect and collect information on residents, so sensors should adapt their operational states according to changes in the resident’s behavior. Second, power management should provide openness to users of ALARM-NET. Users (here users are system administrators) should be able to set different power management actions according to different application situations. Third, power management should be able to control individual functions on the sensor nodes.

For example, a user may want to set a high rate for temperature sensing, a low rate for light sensing, and turn off other sensing functions. Another concern is that ALARM-NET is a heterogeneous sensor network with diverse sensor nodes, such as ECG, motion and other sensors. Power management should be adaptive to control different sensor nodes.

In order to satisfy all these power management requirements, we designed a Context-aware and Open Power Management (COPM) module for ALARM-NET. Many power management schemes in the literature save power using scheduling-based and/or sentry-based algorithms, often to maintain complete sensing coverage for unexpected events [22]. However, these schemes typically are not adjusted automatically or by users at runtime. Since COPM is aware of the individualized patterns of residents, it can dynamically choose power management operations. It also provides flexibility for users to control power management according to peculiarities of the deployment environment. All these properties make COPM a good solution for flexible systems with diverse applications, like ALARM-NET. Specifics of our design for battery-powered motes are now presented.

The COPM module provides users with two types of power management operations—those based on users’ com-

mands and those which are context-aware.

First, users can directly control functions of any mote by sending power management commands. Here, users can turn on/off, or set rates for each individual sensing function in a mote. Also, a user can set the effective period of each command. A typical command may be “Mote No. 1 turns off the light sensor, but senses the temperature every 2 seconds for the following 2 hours.”

Second, the COPM module also provides the context-aware power management operations based on the behavior patterns of the resident. Initially, users can define some context policies for power management, such as “when the resident is sleeping, turn off all sensors in the living room and sense the temperature every 2 minutes in the bedroom.” Based on the resident’s behavior pattern from the CAR, the COPM can trigger the corresponding power management operation when the condition of a context policy is met. This context-aware operation provides more efficient power management that is adaptive to the resident’s behavior. Another advantage of context-aware operation is that the module is open for users to define their own context policies, according to their application demands.

In order to avoid conflicting power management operations, we also attribute a priority to each operation. Here, priorities can be used to differentiate the level of various users.

The COPM module uses four functional components, shown in 9: Sensor Drivers, Context Management, Circadian Activity Rhythm (CAR) and User Interface.

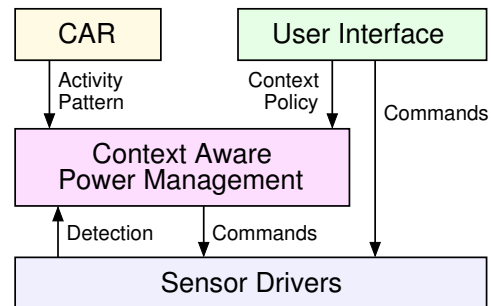


Figure 9. Power Management components in ALARM-NET.

**Sensor Drivers** are used to control the individual functions in motes according to incoming power management operation messages. These messages are generated by a user’s direct commands, or context management. The Sensor controller can turn on/off and set different rates for individual sensing functions, and can also set the duty cycle of a node’s radio components.

A possible problem here is conflicting power management commands. Some commands may want to control the same function with different settings. In order to avoid conflicts, the sensor controller keeps a Sensor State Table (SST). Each entry in SST corresponds to one function of the mote, and it records the current setting (on/off, sensing rate), and the information of the command which set the current state.

Suppose a new command arrives, which will control the light sensor with priority 3. The controller will first look in

the SST and find whether the previous command which set the current state of the light sensor has a higher priority. If so, the new command will be ignored. Otherwise, it will be executed.

**Context Management**, residing in the AlarmGate application, is used to implement context-aware power management operations. This component is the core of COPM. The inputs of the component are context policies defined by users, the sensing reports from motes, and the resident’s behavior patterns obtained from the CAR. The output is a set of power management commands.

We give a concrete example. The user may give a context policy, “turn off all motion sensors when the resident goes to sleep,” and the CAR gives the resident’s activity pattern, “the resident usually sleeps after 10:00 PM.” When the motion sensor reports the resident coming into the bedroom after 10 PM, context management can determine that the resident is sleeping, and sends power management commands to turn off the sensors.

**CAR** is used to provide the behavior pattern of the resident. It collects reports from WSN sensors and predicts the resident’s behavior. With the CAR software, the power management module adapts to the behavior of the resident.

**User Interface** is implemented on PCs which connect to ALARM-NET. It provides users two methods to do the power management. First, users can directly control each sensing function in each sensor in the network. These commands are sent to the sensor nodes via the hierarchical routing protocol. Second, the user can define context policies. With these policies, power management commands are automatically generated by the context management. These policies are sent to the context management module on the AlarmGate from IP clients.

## 9 Data Association

Data association is essential to ALARM-NET. When multiple people share the living facility, data collected by the system must be associated with the right person. This bears similarity to human activity surveillance applications with automated identification. Those application are inherently challenging and typically require powerful sensors with strong tracking and identifying capacity, such as surveillance cameras combined with sophisticated computer vision algorithms [23]. However, deploying video cameras in an assisted-living facility has serious privacy concerns and could be too costly. Biometric-based identification devices such as a finger printer readers or iris scanners have high identification accuracy. However, these sensors are expensive and non-ubiquitous. They require device interactions that are often unnatural or inconvenient for senior residents.

In ALARM-NET, we implement a data association program that uses low cost, heterogeneous, ubiquitous sensors. We increase the utility of sensors and lower the cost by making them “multi-functional.” For example, a scale in front of the freezer serves two purposes: first, it takes the weight measurement, and second, the weight measurement provides information about the identity of the person that uses the freezer. Instead of associating data based solely on physiological data of the user, as many camera based solutions

do, ALARM-NET uses Dempster-Shafer evidential theory to process the combination of location, physiological data, and learned activity patterns.

**Dempster-Shafer** evidential theory is a probability-based data fusion classification algorithm. It is useful when the information source contributing information cannot attribute complete certainty to its output decisions [24].

ALARM-NET tracks each person with coarse granularity, currently at the room level, using low-cost motion sensors. Other sensors are available for this purpose, including infrared tripwire sensors, which are triggered when the user breaks the infrared beam. By deploying two trip wire sensors at the entrance of each room, a user’s traveling direction can be determined and the system can track the number of users in each room. The user’s location can be inferred from “in use” sensors deployed in furniture. For example, the user’s location is determined when the pressure sensor in the sofa is triggered.

Each user’s physiological data is collected beforehand and can be valuable information for data association, especially when there is substantial difference among users. For example, a husband and wife may have a sufficient difference in body weight to give high confidence to data association decisions when either steps onto a floor scale.

ALARM-NET does not require the user to wear a certain body tag all the time, although doing so increases the accuracy of association. ALARM-NET monitors the usage of SATIRE, and associates the data to its owner by default.

The system learns about users’ behavior patterns over time, and uses these patterns to identify the users after the confidence level reaches a certain threshold. The behavior pattern can be a single activity or an activity sequence. ALARM-NET has a learning module that records and examines each tuple of  $\langle \text{time, activity, user} \rangle$  and develops a statistical behavior model.

For each activity performed, a few propositions can be made for who conducted the activity. For example, if only persons A and B are present in a room when a certain activity  $m_1$  happens, two proposition will be made, one for each person. The learning module predicts the probability mass for each person based on history. For example, if the system has recorded 100 occurrences of the same activity, 80 done by A and 20 by B, then the probability mass for A and B with respect to this activity evidence is 80% and 20% respectively.

Formally, if there are  $n$  persons in the living space, regarding event reading  $e$ , and  $P_e(i), i = 1 \dots n$  is the probability of person  $i$  triggered reading  $e$ , then the probability mass regarding event  $e$  is:

$$M_e(i) = \frac{P_e(i)}{\sum_{k=1}^n P_e(k)}$$

Later, if another activity  $m_2$  happens and is determined to be associated with the same person that did activity  $m_1$ , evidence  $m_1$  and  $m_2$  will be combined using Dempster’s rule to identify the person’s identity.

The general form of Dempster’s rule for the total probability mass committed to an event  $c$  defined by the combination of evidence represented by  $m_A(a_i)$  and  $m_B(b_j)$  is given

by

$$m(c) = K \sum_{a_i \cap b_j = c} [m_A(a_i) m_B(b_j)]$$

where  $m_A(a_i)$  and  $m_B(b_j)$  are probability mass assignments and  $K$  is the normalization factor to make all the probability masses sum to unity.

Once the accumulated probability mass assignment for  $c$  is beyond a certain confidence threshold, the system declares  $c$  to be the right association. The system gets self-feedback and thus forms a reinforcement learning process each time an association is done.

Overall, the data association module provides a general framework for system event processing and classification. The learning capability makes it adaptable to changing contexts.

## 10 Implementation

We constructed a testbed to evaluate the overall system and individual components. Here we describe the system implementation, starting with the hardware and sensors used or created.

### 10.1 Hardware and Sensors

ALARM-NET incorporates a heterogeneous hardware environment of sensor, gateway, and user nodes. We use MICAz by Crossbow and Telos Sky by MoteIV as the base for the sensor nodes. They communicate wirelessly among themselves and to the Stargate node made by Crossbow, which performs the AlarmGate role. Users can access ALARM-NET through desktop computers over the Internet or iPAQ PDA or LCD-enabled mote devices wirelessly.

Our system includes commercially available sensors made by Crossbow that measure light intensity, humidity, and acceleration. It also interfaces with ECG and pulse oximeter sensors designed at Harvard by the CodeBlue project [9]. In addition, we have created hardware interfaces to other devices, such as a motion sensor, body scale, blood pressure meter, dust sensor, and LCD module. We will describe these interfaces below.

To detect motion we modified the commercially available wireless motion sensor HawkEye MS13A for X-10 networks. We removed the radio and the controller components, and interfaced the conditioned motion sensor signal to the MICAz 51-pin connector. In addition, the built-in light sensor, LED, and the user interface button were connected to the MICAz. Thus the light sensor and the button can generate interrupts and wake up the node when asleep.

The body scale and blood pressure sensors are attached to the MICAz mote via the serial port. The interface board has a MAX232 interface chip working as a level shifter between RS232 levels and the digital logic levels on the mote.

We built a dust sensor module for MICAz motes. The sensor is based on the SHARP GP2Y1010AU module that has an infrared LED and an amplified photodetector encapsulated in a dark box with air passage. The MICAz mote excites the infrared LED for 0.32ms every 10ms. The dust and smoke particles in the air reflect some of the light to the photosensor that creates electrical impulse responses scaled

and measured by the ADC of the MICAz mote. The air contamination level is related to the amplitude measured by the ADC.

We designed and implemented SeeMote, an LCD module that serves as a user interface device attachable to the MICAz motes. Thus, such an LCD-enabled mote is able to provide visual feedback to medical personnel. For example, it can show the ECG waveform of the patient that has the ECG sensor attached. The LCD has  $128 \times 160$  pixels, each capable of 64K colors. The size of the LCD is 1.8 inches on the diagonal, and it fits on top of the MICAz mote [11]. The module also has five momentary buttons allowing the user to provide input to the system.

All of our hardware designs are open source and available on the website <http://www.openwsn.com>.

### 10.2 Software

All modules shown in Figure 10(a) are implemented in nesC and run on TinyOS 1.1.15 on the MICAz and Telos motes. The TinyOS application `SensorMote` wires them together along with modules for routing, configuration, and localization not described here. Compiler flags define the particular sensor drivers (see Figure 3) included in the application.

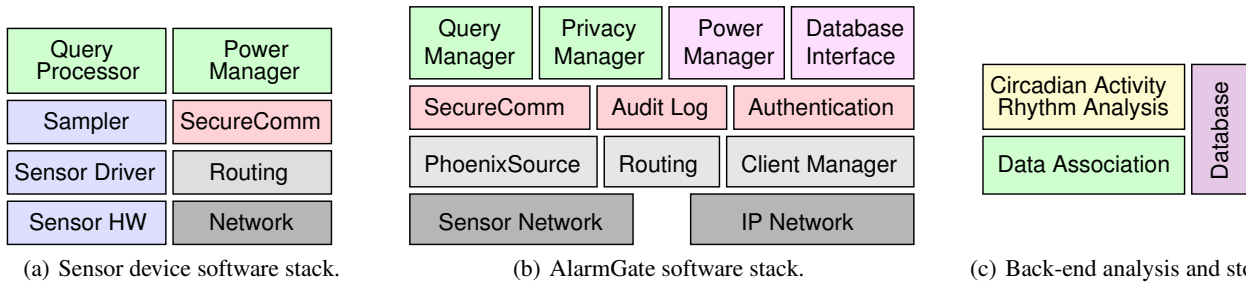
Modifications to the TinyOS timer module was necessary for the `SensorSamplerC` component. This part of the `SamplerC` configuration tracks the next scheduled sample using a single timer, and performs the sample on expiry. After the first of multiple queries is received, the module needs to know either how much time has elapsed since the timer was started, or how much time remains. This functionality is not exposed in the original `Timer` interface.

The `SecureComm` components required significant modification to TinyOS provided code, since inline cryptography must be managed while messages are being transferred to/from the TX/RX FIFOs of the radio chip. As described in Section 7 and shown in Figure 8, `CC2420SecurityC` must insert security configuration while messages are being received or before they are transmitted. Routines for reading partial RXFIFO contents and writing to CC2420 RAM were created.

The `AlarmGate` application is written in Java and uses the 1.3.1 API in order to be supported by the Blackdown Java2 v1.3.1 JRE for ARM-Linux. The open-source `RXTX` library was used for access to serial communication from Java. Files were stored on an external `CompactFlash` expansion card.

Application-level modules shown in Figure 10(b) have been described. In addition, some lower-level modules handle connection to the WSN and IP networks. `PhoenixSource` is distributed with TinyOS, and provides a connection to the stargate-attached MICAz mote through a serial connection. It was modified to use the `gnu.io` namespace rather than `javax.comm`, for use with the `RXTX` library. An IP server socket listens in the `Client Manager`, spawning new threads to handle connecting clients.

A Java library was created to contain code common to `AlarmGate` and clients. It contains classes for message handling, query and report parsing, SRP authentication, secure communication, and database access. The `Database Interface` uses `JDBC MySQL connector 3.0.17` to connect to the



**Figure 10. ALARM-NET component software. Sensor code is implemented in nesC. AlarmGate and custom back-end software is Java and C.**

back-end database. The freely available Lightweight Bouncy Castle API v1.30 [25] was used for SecureComm cryptography modes implementations (CBC-MAC, CTR, CCM).

Back-end programs (see Figure 10(c)) reside on one or more PCs connected to the same IP network as the stargates. A standard installation of MySQL 5.0 maintains the system configuration, including: sensor types, locations, and owners; and, user information, roles, and privacy policies. Sensor data, audit logs, and alerts are stored in the database by the AlarmGate for consumption by administrators or back-end analysis by the CAR and Data Association modules.

Circadian Activity Rhythm (CAR) Analysis is performed by a NI LabWindows/CVI application with GUI. Data are collected and analyzed automatically, updating activity patterns and sending occupancy reports and deviation alerts to the AlarmGate.

The Data Association program is currently implemented using Java 1.5 and runs on a back-end PC. It connects to the database using JDBC connection and polls the database every second to process newly arrived sensor data.



**Figure 11. PDA Query Issuer.**

Reports are graphed in real-time as they stream in from the network.

## 11 System Evaluation and Performance

To evaluate ALARM-NET’s performance, we built the system as describe above. Since its operation spans multiple functional components, we present a series of scenarios to confirm correct behavior then include quantitative measurements of important components as space permits.

### 11.1 Integration and Back-End Analysis

In the first scenario, we tested a seven-room assisted-living residential unit (see Figure 12), with a motion sensor on each wall, a PC running the back-end, and a stargate with the AlarmGate application.

A person representing the resident moved through the living space from room to room, sometimes pausing with no activity. We verified that the motion sensors reliably detected the person’s movement, and that reports were collected by AlarmGate and stored in the back-end database.

The CAR application, also connected to the database, received and processed the reports, comparing with the current behavioral trend. Since the movements did not represent deviations, context messages sent to the Privacy Manager on the AlarmGate indicated a normal status.

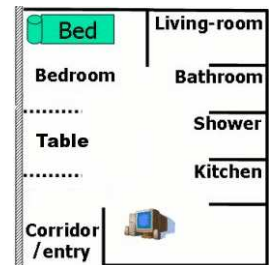
A second person entered the living space. Both residents continued to move periodically, though they maintained disjoint trajectories. This motion data was collected from the database by the data association program on the back-end, which correctly maintained unique identity assignments for the two concurrent sets of readings. This second person then left the unit.

### 11.2 Privacy-Aware Queries

Using the same testbed configuration as in scenario 1, we added a wearable pulse rate sensor to the resident and an environmental temperature sensor to the living room. The resident’s CAR behavior deviation continues to be normal.

A user of the system, a technician, connects to the AlarmGate and issues a single query for the environmental temperature of the resident’s unit. The Query Manager on the AlarmGate consulted with the Privacy Manager for authorization. Since the resident’s privacy policy for a technician for temperature is “DF,IHT”, or default deny, the query was not authorized. The Query Manager sent a NACK to the technician to indicate the denial.

The simulated resident then moved around the living space in a manner different from the behavior profile learned



**Figure 12. Simulated Residence.**

by the CAR. The context for the resident was updated by a CAR alert sent to the Privacy Manager.

Another person with a PDA, representing a nurse, responded to the CAR deviation. The nurse connected to AlarmGate using the PDA, and issued a periodic query for the pulse rate of the resident. As before, the Query Manager authorized the request with the Privacy Manager. Though the resident's default policy for nurse access to pulse data was deny, the heightened context alert value for the resident permitted the query.

The Query Manager determined the proper sensor in the WSN to receive the query, and forwarded it. Upon receipt of the query, the pulse rate sensor streamed readings back to the PDA (via the AlarmGate) at the requested report rate, where it was graphed in real-time as it arrived.

Satisfied that the resident was okay, the nurse stopped the real-time query and left the residential unit.

### 11.3 Context-Aware Power Management

In the previous scenarios, the CAR application also sent occupancy information to the Power Management module on the AlarmGate, which controls sensors according to context and policy.

To analyze the efficiency of our power management scheme under different contexts and policies, we first collect energy use of the motes and sensors, then calculate power savings for a subset of rooms in the previous scenarios: the bedroom, kitchen and living room. In each room, we have one mote with a motion sensor called M-mote, and one mote for sensing environmental data, called E-mote. A single resident lives in the unit and wears a body network with one mote (B-mote).

We first measured the power supply voltage and current of the whole mote with one functional component disabled at a time by the software. Here, we use the MICAz mote with MTS310 sensor board which can provide 2-axis accelerometer, 2-axis magnetometer, light, temperature, and acoustic sensing modalities. While the measured voltage was constant at 3V for the fresh pair of AA batteries, the current was as shown in Figure 13(a).

For this analysis, we classified certain combinations of sensors that can be used in ALARM-NET to define the following four typical power modes:

- **Environment Sense (ES):** A mote in ES enables the light, temperature and acoustic sensing components, as well as the radio to send the sensing report.
- **Body Sense (BS):** A mote in BS is typically involved in a body network. It should enable the accelerometer to report the resident's movement and the radio.
- **Standby:** A mote in Standby has the Radio on and ready to receive or transmit data, but all the sensors are turned off.
- **Sleep:** When the mote is in the sleep mode, the radio and all the sensors are turned off.

Next, we measured the supply current of the mote when in these four different power modes, presented in Figure 13(b).

We also measured the power consumption of the motes with a motion sensor, shown in Figure 13(c). Here, a good policy is to always enable the motion sensor and disable the

radio, and let the interrupt from the motion sensor wake up the radio to send the report.

After defining some basic context policies, we calculate estimated power expended with and without our power management:

- Policy 1: For M-motes, keep the motion sensor on and the radio off until a motion interrupt turns it on.
- Policy 2: When the resident is sleeping, all the E-motes and the B-mote switch to the Sleep mode.
- Policy 3: When the resident is awake, the B-mote should stay in the BS mode, the E-motes in the room where the resident is staying switch to the ES mode, and all other E-motes go to Standby mode.

Suppose that the resident sleeps for an average of 8 hours a day. We can calculate the daily current consumed in the system without any power management. Here, we assume that M-motes also apply Policy 1, and all E-motes and B-mote stay in the ES and BS modes in the whole day. By calculation, the daily current consumption in the system is 3753.6 *mAh*.

For ALARM-NET with context-aware power management, we assume that the CAR software can always predict the resident's behavior correctly for the purpose of this analysis. So, E-motes will stay in the Sleep modes for 8 hours, in the ES mode for  $\frac{16}{3}$  hours and in the Standby state for  $\frac{32}{3}$  hours. B-motes will stay in the sleep mode for 8 hours, and in the BS mode for 16 hours. Adding the consumption of M-motes, the daily current consumption is 2324.8 *mAh*, which is only 61.9% of the previous consumption.

### 11.4 Secure Networks

In our test scenarios, the technician and nurse connecting to the AlarmGate had to provide the proper authenticating credentials (username and password) before issuing queries. The nurse entered the wrong password at first, and was denied access to the network.

To further evaluate the cost of securing messages in the IP and sensor networks, we measured the message and processing overheads. The messages are always longer than without security, and it takes longer to perform cryptographic operations, hence security always comes at a cost. For applications that require security, as ALARM-NET does, the goal is to minimize the overhead cost of security.

Figure 14(a) shows percentage overhead calculations due to message length expansion. The overhead due to the SecureComm header and footer (MAC) naturally decreases as the payload length increases. A vertical line shows the default payload length of 29 bytes in the WSN. In ALARM-NET we use a length of 70 bytes (out of a maximum of 116) in the WSN, and 255 in the IP network.

When receiving a message on the mote, SecureComm reads enough of the RXFIFO to determine whether this message should be processed as a secure frame, and in what mode. The inline security operation is configured on the CC2420 and the remainder of the frame is read out of the RXFIFO. As the message propagates through the SecureComm component, the header and footer are stripped out of the payload before it is signalled to the higher layers.

The processing time for all these steps is shown in Figure

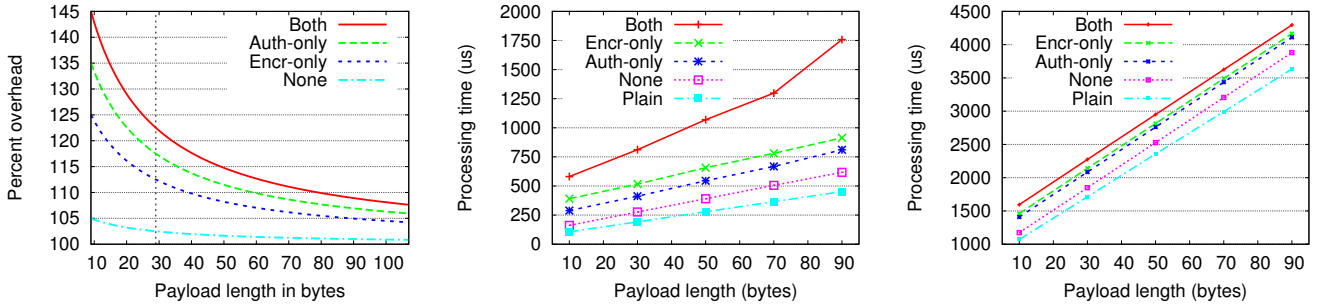
Components	Supply Current	Power Modes	Supply Current	Motion Sensor	Supply Current
All enabled	45.0 mA	ES	33.9mA	All components enabled	32.5mA
Temp disabled	44.9	BS	33.4mA	Motion sensor disabled	32.5mA
Light disabled	44.9	Standby	32.8mA	Radio disabled	7.1mA
Accel disabled	44.4	Sleep	6.2mA		
Magnet disabled	35.1				
Acoustic disabled	44.0				
Radio disabled	19.2				

(a) Power Measurement with disabled components.

(b) Power Measurement in different power modes.

(c) Power Measurement of motion sensor.

**Figure 13. Power Measurements for evaluation of context-aware power management in ALARM-NET.**



(a) Message overhead compared to standard messages. Default payload length of 29 bytes is shown.

(b) Message receive processing time. Excludes task wait time.

(c) Message transmit processing time. Excludes task wait time.

**Figure 14. SecureComm overhead and performance.**

14(b). The time required for receiving an insecure (“plain”) message is also shown. Multiple asynchronous task postings make simple measurement of the processing time difficult. The data presented here are the sums of the synchronous code blocks, so that task dynamics do not obscure the measurements. Times were measured using the Tektronix TDS 2022 scope’s rise-fall measure function. Also, times do not include one-time costs like copying the key to the radio chip. The largest incremental cost comes when using both encryption and authentication together.

Finally, Figure 14(c) shows similar measurements for the transmit path. These times include shifting the payload to make room for the SecureComm header, setting up transmit security, writing the frame to the TXFIFO, and cleaning up. In contrast to the receive measurements, these include the actual radio transmission time, and so are uniformly higher than Figure 14(b).

Application	Lines	Code Size	Data Size
SensorMote	7224	23158	2031
SensorMote-alone <sup>†</sup>	7224	13334	1638
AlarmGate	7429	1342241	—
PDA Query Issuer	8210	340950	—

**Table 1. Code sizes (lines of code, code and data memory in bytes) for applications in ALARM-NET. Java applications include all third-party jar files.**

## 11.5 Embedded Code Details

Table 1 shows the size of code developed for ALARM-NET, in lines of code, program memory, and data memory. We see that the SensorMote TinyOS application easily fits into the motes’ code flash (maximum size 128KB).

Data memory for this application depends primarily on the TinyOS message length and the maximum size of state tables in the QueryProcessor and Sampler modules. The measurements in Table 1 were taken with a 70-byte message payload, and a maximum of five concurrent queries, samples, and keys in the KeyStore.

SensorMote-alone<sup>†</sup> shows the code and data used beyond that required for the simple CntToRfm<sup>1</sup> application, to show how many resources are used by our application and its modules vs. those in TinyOS itself.

Java code for the PDA and stargate easily fit within their memory constraints. Code sizes given are the sizes of the class files, including the JDBC connector, crypto library, and PhoenixSource and its dependences.

## 12 Conclusion

We presented ALARM-NET, a wireless sensor network designed for long-term health monitoring in the assisted-living and residential environments. A central design goal was to adapt the operation of the system, including power management and privacy policy enforcement, to the individual life patterns of the residents, which are analyzed and fed

<sup>1</sup>CntToRfm increments a counter every 250ms and transmits its value in a radio message—perhaps the simplest application that uses basic TinyOS components like Timer and Radio.

back into the system. This is only possible if we can associate data with the proper individual, for which we use sensor data and Dempster-Shafer evidential theory. Protection of medical information is very important, so IP and WSN communication is secured with SRP and SecureComm, a hardware accelerated security suite for Chipcon CC2420-based devices.

### 13 References

- [1] President's Information Technology Advisory Committee (PITAC), Revolutionizing Health Care Through Information Technology, June 2004. URL: [http://www.nitrd.gov/pitac/reports/20040721\\_hit\\_report.pdf](http://www.nitrd.gov/pitac/reports/20040721_hit_report.pdf).
- [2] T. He, S. Krishnamurthy, L. Luo, T. Yan, B. Krogh, L. Gu, R. Stoleru, G. Zhou, Q. Cao, P. Vicaire, J. A. Stankovic, T. F. Abdelzaher, and J. Hui. VigilNet: An integrated sensor network system for energy-efficient surveillance. *ACM Transaction on Sensor Networks*, to appear, 2006.
- [3] G. Werner-Allen, J. Johnson, M. Ruiz, J. Lees, and M. Welsh. Monitoring volcanic eruptions with a wireless sensor network. In *Proc. European Workshop on Sensor Networks (EWSN'05)*, January 2005.
- [4] S. Katz, A. B. Ford, R. W. Moskowitz, B. A. Jackson, and M. W. Jaffe. Studies of illness in the aged. the index of adl: A standardized measure of biological and psychosocial function. *Journal of the American Medical Association*, 185:914–919, September 1963.
- [5] Intel. Digital home technologies for aging in place. URL: [http://www.intel.com/research/exploratory/digital\\_home.htm](http://www.intel.com/research/exploratory/digital_home.htm).
- [6] New York University of Rochester. Center for future health – smart medical home. URL: [http://www.futurehealth.rochester.edu/smart\\_home](http://www.futurehealth.rochester.edu/smart_home).
- [7] Cory D. Kidd, Robert J. Orr, Gregory D. Abowd, Christopher G. Atkeson, Irfan A. Essa, Blair MacIntyre, Elizabeth Mynatt, Thad E. Starner, and Wendy Newstetter. The aware home: A living laboratory for ubiquitous computing research. In *Proceedings of the Second International Workshop on Cooperative Buildings*, 1999.
- [8] MIT (Massachusetts Institute of Technology). House\_n: the home of the future. URL: [http://architecture.mit.edu/house\\_n](http://architecture.mit.edu/house_n).
- [9] D. Malan, T. Fulford-Jones, M. Welsh, and S. Moulton. Codeblue: An ad hoc sensor network infrastructure for emergency medical care. In *Proceeding of the International Workshop on Wearable and Implantable Body Sensor Networks*, 2004.
- [10] University of Washington. The assistive cognition project. URL: <http://www.cs.washington.edu/assistcog>.
- [11] L. Selavo, G. Zhou, and J.A. Stankovic. Multimodal user interface and power meter module for wireless sensor networks. Submitted to The 3rd Annual International Conference on Mobile and Ubiquitous Systems: Networks and Services (MOBIQUITOUS 2006).
- [12] Raghu K. Ganti, Praveen Jayachandran, and Tarek F. Abdelzaher. SATIRE: A software architecture for Smart AtTIRE. In *4th ACM Conference on Mobile Systems, Applications, and Services (Mobisys 2006)*, 2006.
- [13] G. Virone, M. Alwan, S. Dalal, S. Kell, J. A. Stankovic, and R. Felder. Behavioral patterns of older adults in assisted living, 2006. submitted to IEEE Transactions on Biomedical Engineering, 2006.
- [14] T. Wu. The secure remote password protocol. In *Proceedings Network and Distributed System Security Symposium*, pages 97–111, March 1998.
- [15] National Institute of Standards and Technology (NIST). *FIPS PUB 197, Specification for the Advanced Encryption Standard (AES)*, November 2001.
- [16] IEEE 802.15.4-2003 IEEE Standard for Information technology. *Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (LR-WPANs)*, 2003.
- [17] National Institute of Standards and Technology (NIST). *FIPS PUB 113, Standard on Computer Data Authentication*, May 1985.
- [18] Whitfield Diffie and Martin Hellman. Privacy and authentication: An introduction to cryptography. *Proceedings of the IEEE*, 67:397–427, 1979.
- [19] D. Whiting, R. Housley, and N. Ferguson. *RFC 3610 – Counter with CBC-MAC (CCM)*, IETF. IETF, 2003.
- [20] Adrian Perrig, Robert Szewczyk, Victor Wen, David Culler, and J. D. Tygar. SPINS: Security protocols for sensor networks. In *Proceedings of Seventh Annual International Conference on Mobile Computing and Networks MOBICOM 2001*, pages 189–199, July 2001.
- [21] Chris Karlof, Naveen Sastry, and David Wagner. TinySec: A link layer security architecture for wireless sensor networks. In *Second ACM Conference on Embedded Networked Sensor Systems (SenSys 2004)*, November 2004.
- [22] D. Tian and N. D. Georganas. A node scheduling scheme for energy conservation in large wireless sensor networks. In *Wireless Communications and Mobile Computing*, 2003.
- [23] N. Atsushi, K. Hirokazu, H. Shinsaku, and I. Seiji. Tracking multiple people using distributed vision systems. In *Robotics and Automation*, volume 3, pages 2974–2981, 2002.
- [24] Lawrence A. Klein. *Sensor and Data Fusion, A Tool for Information and Decision Making*. SPIE, July 2004.
- [25] Bouncy Castle Crypto APIs, URL: <http://www.bouncycastle.org/>.