Sensor Networks for Medical Care

Victor Shnayder, Bor-rong Chen,
Konrad Lorincz, Thaddeus R. F. Fulford-Jones,
and Matt Welsh

TR-08-05

Computer Science Group
Harvard University
Cambridge, Massachusetts

Sensor Networks for Medical Care

Victor Shnayder, Bor-rong Chen, Konrad Lorincz,
Thaddeus R. F. Fulford-Jones, and Matt Welsh
Division of Engineering and Applied Sciences
Harvard University
{shnayder,brchen,konrad,mdw}@eecs.harvard.edu, fulford@fas.harvard.edu

Abstract

Sensor networks have the potential to greatly impact many aspects of medical care. By outfitting patients with wireless, wearable vital sign sensors, collecting detailed real-time data on physiological status can be greatly simplified. However, there is a significant gap between existing sensor network systems and the needs of medical care. In particular, medical sensor networks must support multicast routing topologies, node mobility, a wide range of data rates and high degrees of reliability, and security.

This paper describes our experiences with developing a combined hardware and software platform for medical sensor networks, called *CodeBlue*. CodeBlue provides protocols for device discovery and publish/subscribe multihop routing, as well as a simple query interface that is tailored for medical monitoring. We have developed several medical sensors based on the popular MicaZ and Telos mote designs, including a pulse oximeter, EKG and motion-activity sensor. We also describe a new, miniaturized sensor mote designed for medical use.

We present initial results for the CodeBlue prototype demonstrating the integration of our medical sensors with the publish/subscribe routing substrate. We have experimentally validated the prototype on our 30-node sensor network testbed, demonstrating its scalability and robustness as the number of simultaneous queries, data rates, and transmitting sensors are varied. We also study the effect of node mobility, fairness across multiple simultaneous paths, and patterns of packet loss, confirming the system's ability to maintain stable routes despite variations in node location and data rate.

1 Introduction

An emerging application for wireless sensor networks involves their use in medical care. In a hospital or clinic, outfitting every patient with tiny, wearable wireless vital sign sensors would allow doctors, nurses and other caregivers to continuously monitor the status of their patients. In an emergency or disaster scenario, the same technology would enable medics to more effectively care for large numbers of casualties. First responders could receive immediate notifications on any changes in patient status, such as respiratory failure or cardiac arrest. Wireless sensors could augment or replace existing wired telemetry systems for many specific clinical applications, such as physical rehabilitation or long-term ambulatory monitoring.

Despite the increased interest in this area, a significant gap remains between existing sensor network designs and the requirements of medical monitoring. Most sensor networks are intended for deployments of stationary nodes that transmit data at relatively low data rates, with a focus on best-effort data collection at a central base station. By contrast, medical monitoring requires relatively high data rates, reliable communication, and multiple receivers (e.g. PDAs carried by doctors and nurses). Moreover, unlike many sensor network applications, medical monitoring cannot make use of traditional in-network aggregation since it is not generally meaningful to combine data from multiple patients.

This paper presents our initial experiences with a prototype medical sensor network platform, called *CodeBlue*. We have developed a range of medical sensors integrated with the commonly-used Mica2 [8], MicaZ [9] and Telos [41] mote designs. These include a pulse oximeter [37], two-lead electrocardiogram (EKG) [17], and a specialized motion-analysis sensor board. In addition, we have developed a small form factor variant of the Telos mote specifically for wearable use.

The CodeBlue software framework provides protocols for device discovery, publish/subscribe multihop routing, and a simple query interface allowing caregivers to request data from groups of patients. In addition to monitoring patient vital signs, CodeBlue also integrates an RF-based localization system, called MoteTrack [34], to track the location of patients and caregivers. This capability is especially valuable in large hospital settings. We present an initial evaluation of the CodeBlue prototype, demonstrating its scalability and robustness as the data rates, number of simultaneous queries, and transmitting sensors are varied. We also study the effect of node mobility, fairness across multiple simultaneous paths, and patterns of packet loss, confirming the system's ability to maintain stable routes despite variations in node location and data rate.

We are collaborating with several hospitals and medical research groups that plan to make use of the CodeBlue platform. These include Boston Medical Center, Brigham and Women's Hospital, the Spaulding Rehabilitation Hospital, and Johns Hopkins University. We present initial results demonstrating our wireless motion-analysis sensors, which will be used in a future study of stroke patient rehabilitation. Our initial experience highlights a number of open challenges facing the adoption of low-power wireless sensors for medical deployments. These challenges include effective congestion management, reliable networking, and security.

In the following section we present background on medical sensor networks and discuss related work. In Section 3 we detail our medical sensor hardware designs. Section 4 describes the CodeBlue protocol architecture and prototype implementation. In Section 5 we present initial results evaluating the performance of the CodeBlue system on our 30-node indoor sensor network

This document is a technical report. It should be cited as:
Technical Report TR-08-05, Division of Engineering and Applied Sciences, Harvard University, 2005.
For more information on this project, please see: `http://www.eecs.harvard.edu/~mdw/proj/codeblue`

testbed. Finally, Section 6 discusses future work and concludes.

2 Motivation and Background

Medical care is an oft-cited application for sensor networks [29, 11]. The ability to augment medical telemetry with tiny, wearable, wireless sensors would have a profound impact on many aspects of clinical practice. Emergency medical care, triage, and intensive care can all benefit from continuous vital sign monitoring, especially immediate notification of patient deterioration. Sensor data can be integrated into electronic patient care records and retrieved for later analysis. In a wide range of clinical studies, especially those involving ambulatory or at-home monitoring, wireless sensors would permit data acquisition at higher resolution and for longer durations than existing monitoring solutions.

Wireless medical telemetry is not altogether new. A number of wireless medical monitors are currently on the market, including electrocardiographs (EKGs) [21, 23, 18], pulse oximeters [42, 50], blood pressure monitors [6, 1], and fetal heart rate and maternal uterine monitors [19]. Most of these devices use Bluetooth or the analog Wireless Medical Telemetry Service (WMTS) bands [16], although several employ IEEE 802.11. However, these systems are generally designed only to "cut the cord" between the sensor worn by the patient and a bedside monitor or other nearby receiving device. They are not intended to participate in a network, to relay data to multiple receivers (e.g. by means of multi-hop routing), or to scale to large numbers of monitors in an area. In addition, few of these systems are designed to be wearable; most remain attached to the hospital bed, and the few wireless ambulatory products on the market are generally large and cumbersome. As an example, the Welch-Allyn Micropaq monitor [50] measures over 18 cm $\times$ 8.8 cm $\times$ 4cm and weighs nearly half a kilogram.

The emergence of low-power, single-chip radios based on the Bluetooth and 802.15.4 [26] standards has precipitated the design of small, wearable, truly networked medical sensors. In a mass casualty or disaster setting, medics can place tiny sensors on each patient to form an *ad hoc* network, relaying continuous vital sign data to multiple receiving devices (e.g. PDAs carried by physicians, or laptop base stations in ambulances). In addition to relaying vital sign data, each node can act as an "active triage tag," storing information about the wearer (identification, medical history, severity status, etc.) RF-based localization can be used to track patient and first responder location on the scene. Such a system can be translated directly into hospital settings where wired monitoring is cumbersome and (especially with pediatric and neonatal patients) obstructs the caregiver's access to the patient.

2.1 Requirements

The requirements for a medical sensor network design depend greatly on the specific application and deployment environment. A sensor network designed for *ad hoc* deployment in an emergency situation has very different requirements than one deployed permanently in a hospital. For example, the latter can make use of fixed, powered gateway nodes which provide access to a wired network infrastructure. In general, however, we can identify several characteristics that nearly all medical sensor networks would share.

Wearable sensor platforms: Medical applications generally require very small, lightweight, and wearable sensors. Existing mote platforms are good for demonstrations, but we have found that the large battery packs and protruding antennas are suboptimal for medical use.

Reliable communications: In medical settings, a great emphasis is placed on data availability. Although intermittent packet loss due to interference may be acceptable, persistent packet loss (due to congestion or node mobility) would be problematic. Depending on the sensors in use, sampling rates may range anywhere from less than 1 Hz to 1000 Hz or more, placing heavy demands on the wireless channel.

Multiple receivers: We expect that the data from a given patient will typically be received by multiple doctors or nurses caring for the patient. This suggests that the network layer should support *multicast* semantics.

Device mobility: Both patients and caregivers are mobile, requiring that the communication layer adapt rapidly to changes in link quality. For example, if a multihop routing protocol is in use, it should quickly find new routes when a doctor moves from room to room during rounds.

Security: Aside from the obvious security considerations with sensitive patient data, United States law mandates that medical devices meet the privacy requirements of the 1996 Health Insurance Portability and Accountability Act (HIPAA). Recent work on private-key and public-key cryptography schemes for sensor networks [29, 22, 38] is applicable here, but must be integrated into an appropriate authentication and authorization framework.

2.2 Related work

Many of the aforementioned requirements have not yet been adequately addressed by the sensor network community. The chief reason is that most sensor network applications have very different data, communication, and lifetime requirements. Unlike traditional data collection applications such as environmental monitoring [7, 48, 51], medical deployments are characterized by mobile nodes with varying data rates and few opportunities for in-network aggregation. In addition, medical sensor networks are less concerned with maximizing individual node lifetimes, since it is acceptable to recharge devices or change batteries on a relatively frequent basis.

As a result, many of the significant advances in communication models [27, 53], time synchronization [39, 15], and energy management [44] should be reevaluated given these new requirements. This is not to say that we must start from scratch; rather, we believe it is best to borrow from prior systems as much as possible and invent new technology only as needed.

A number of other research projects are exploring medical sensor networks. Most of these projects are concerned with developing wearable medical sensors [33, 54, 46], while others have developed infrastructures for monitoring individual patients during daily activity [30], at home [12], or at a hospital [31]. In contrast, our focus is to develop a robust, scalable infrastructure for deploying sensor networks in a range of medical settings.

More closely related to our efforts are systems for enabling large numbers of medical sensors to be used for disaster response. The SMART [43], AID-N [52], and WiiSARD [32] teams are among several funded through a US National Library of Medicine effort to develop new technologies for disaster management. The AID-N group is making use of our sensor designs, and the SMART team has developed a mote-based EKG [46] that is largely equivalent to our design described in Section 3. The WiiSARD group has developed a prototype pulse oximeter based on an 802.11-equipped PDA, but its size and power requirements make it impractical for real medical use. The WiiSARD and SMART designs call for a central server to collect and distribute all sensor data, an approach with obvious reliability and scalability considerations. We are not aware of any published material describing the communication, routing, discovery, or data query mechanisms used by these systems.

3 Wireless Medical Sensors

Medical applications of sensor networks require new hardware designs. In this section we detail three mote-based medical sensors that we have developed: a mote-based pulse oximeter, two-lead electrocardiograph (EKG), and a special-purpose motion-analysis sensorboard. We also describe Pluto, our custom mote design for wearable applications.

3.1 Pulse oximeter

Pulse oximetry has been in use as a medical diagnostic technique since its invention in the early 1970s [49]. This non-invasive technology is used to reliably assess two key patient health metrics: heart rate (HR) and blood oxygen saturation (SpO_2). These parameters yield critical information, particularly in emergencies when a sudden change in the heart rate or reduction in blood oxygenation can indicate a need for urgent medical intervention. Pulse oximetry can provide advance warning of the onset of hypoxemia even before the patient manifests physical symptoms.

3.1.1 Technology

Pulse oximetry involves the projection of infrared and near-infrared light through blood vessels near the skin. Pulse oximeters typically incorporate a plastic housing that slips over the index finger or earlobe. The housing contains an array of LEDs along one inner surface and an optoelectronic sensor opposite.

By detecting the amount of light absorbed by hemoglobin in the blood at two different wavelengths (typically 650nm and 805nm), the level of oxygen saturation can be measured. In addition, heart rate can be determined from the pattern of light absorption over time, since blood vessels contract and expand with the patient's pulse. Computation of HR and SpO_2 from the light transmission waveforms can be performed using standard digital signal processing (DSP) techniques. Sophisticated algorithms have been developed to mitigate errors due to motion artifacts [45].

3.1.2 Mote-based pulse oximeter

In developing a mote-based pulse oximeter, we were fortunate that there exist several available products that provide self-contained logic for driving the LEDs and performing the HR and SpO_2 calculations. We initially considered the Dolphin Medical [13] OEM 601 and 701 units, credit-card sized boards that contain all of the required signal processing logic. However, they did not meet our requirements due to a current consumption of over 100 mA and an operating voltage of 5 V.

The smallest and lowest-power OEM module that we are aware of is the BCI Medical Micro-Power Pulse Oximeter [47], measuring 39 mm $\times$ 20 mm with a current draw of just 6.6 mA at 3 V. The board performs all of the required calculations and relays vital sign data over a serial line which can be readily interfaced to a mote. The board reports heart rates in the range 30–254 bpm and SpO_2 values from 0 to 99%.

Our pulse oximetry sensor board is essentially a connector between the Mica2/MicaZ mote platform and the BCI Medical board (see Figure 1(a)). Our board incorporates the MicaZ's 51-pin connector, the two headers for the BCI board, and a DB9 connector for the finger sensor. A TinyOS module on the mote controls the BCI hardware (which can be reset using two digital I/O pins from the mote) and parses the serial protocol to determine HR and SpO_2. When powered on, the BCI module requires about 20 seconds to acquire the waveform and report vital sign data. If the finger sensor is detached from the patient, the board reports an error condition using out-of-range vital sign values.

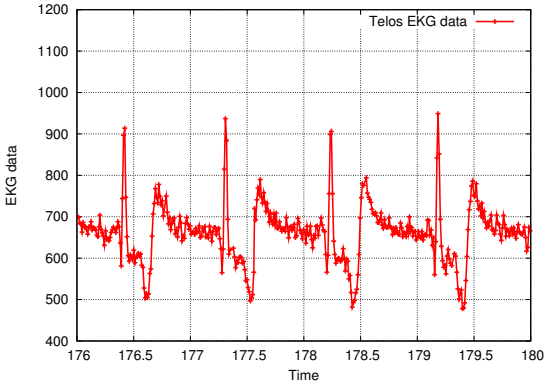


Figure 2: **Sample EKG trace captured from our Telos-based EKG sensor board.**

3.2 Electrocardiograph (EKG)

Two different types of electrocardiograph (EKG or ECG) are commonly used in clinical and trauma care to measure the electrical activity of the heart. The most prevalent EKG type involves the connection of between twelve and fifteen leads to a patient's chest, arms and right leg via adhesive foam pads. The device records a short sampling (not more than thirty seconds) of the heart's electrical activity between different pairs of electrodes. Each pair of leads provides a unique and detailed picture of the cardiac rhythm, an individual echo of the heart's electrical impulses as they are conducted through surrounding tissue. An experienced cardiologist can rapidly interpret a standard EKG tracing to diagnose a wide range of cardiac arrhythmias, as well as acute myocardial ischemia and infarction.

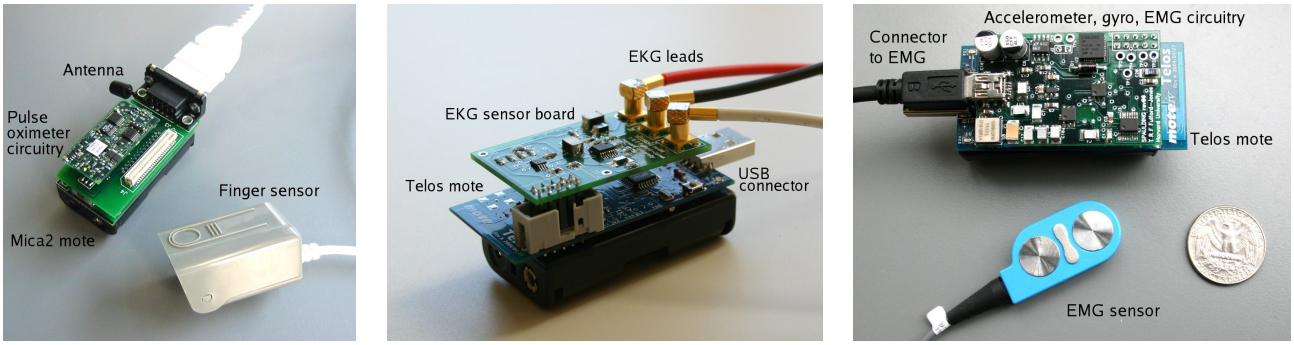
However, because standard EKG traces only represent a short sampling of patient data, irregular or intermittent cardiac conditions may not be identifiable. To address this shortcoming, many hospitals also employ *continuous EKG telemetry* to monitor patients in intensive care. This involves the use of a two- or three-electrode EKG to evaluate a patient's cardiac activity for an extended period. The amplified heart signals are either displayed on a screen or printed onto a roll of paper adjacent to the patient's bedside. A physician may advise continuous monitoring if there is a chance that a patient has cardiac problems, such as arrhythmia, that occur intermittently, maybe only once or twice a day. Continuous telemetry may also be useful as a means of alerting healthcare staff to the first signs of deterioration of a patient's condition.

EKG systems operate by acquiring and amplifying the electrical signals generated with each contraction and expansion of the cardiac muscle. Commercial systems generally incorporate one or more instrumentation amplifiers with excellent common mode noise rejection and signal amplification characteristics. In addition, such systems may include dedicated signal processing circuitry to further enhance the quality of the tracing.

Many EKG machines, both standard and continuous, are marketed as "portable" but this does not necessarily mean that they are small and unobtrusive. Most such appliances receive power from an electrical outlet and are sufficiently heavy that they must be mounted on a cart and wheeled from one location to the next.

3.2.1 Mote-based EKG

We have developed sensor boards for both the Mica2/MicaZ and Telos mote platforms that provide continuous EKG monitoring by measuring the differential across a single pair of electrodes (see Figure 1(b)). The circuit design incorporates the Texas Instruments INA321 CMOS instrumentation amplifier; an earlier version of the schematic is shown in [17]. More recent revisions of



(a) Pulse oximeter **(b) EKG** **(c) Motion capture and EMG**

Figure 1: **Wireless medical sensors developed by our group.**

the circuit include additional operational amplifiers together with a multitude of passive components to provide enhanced filtering. The circuit draws power from the mote's battery pack.

Connectors are provided to three leads that attach to the patient's upper and lower chest; one lead serves to properly bias the patient's skin while the other two are used to measure cardiac activity. The INA321 amplifies the differential signal by a factor of 5 and filters out almost all common-mode noise. A high-pass feedback filter dynamically corrects any DC shift that may occur over time. The signal subsequently passes to an op-amp that provides further amplification and acts as a low-pass filter. The resulting trace is routed to an ADC port on the mote. A TinyOS component samples the EKG signal at a configurable frequency (typically 120 Hz). A sample trace captured from the Telos-based EKG sensor is shown in Figure 2. Evaluation of the circuit's output by a clinical cardiologist confirmed that it was comparable to that of commercial EKG machines.

3.3 Motion analysis sensor board

Apart from traditional vital sign monitoring, sensor networks can be used in specific clinical studies that may require specialized instrumentation to record physiological signals of interest. We are working with researchers at Spaulding Rehabilitation Hospital in Boston to develop wireless sensors for two studies involving motion analysis. The first focuses on patients undergoing physical rehabilitation after stroke while the second aims to evaluate the effectiveness of treatments for Parkinson's Disease. Both studies require capturing detailed data on muscular activity and on limb movements.

Stroke is a form of brain damage caused either by internal bleeding or by an acute lack of blood in some part of the brain. In either case, the function of a part of the brain is temporarily or permanently stopped. A recovering stroke patient may experience impaired movement and weakness of one half of the body, in addition to speech problems and difficulty maintaining a sense of balance.

Parkinson's Disease is a degenerative brain disorder that typically develops after the age of 50. The characteristic symptom of Parkinson's is an involuntary and uncontrollable shaking (called tremor) that usually starts in the hands but which, if left untreated, can eventually spread throughout the body. In many cases the cause is unclear, but Secondary Parkinson's Disease may be triggered by conditions such as brain injury or certain brain infections. More accurate measurement of motor fluctuations during daily life would benefit patients by enabling doctors to fine-tune the dosage and timing of existing medications such as levodopa. It would also help researchers to better evaluate new therapies in clinical trials.

3.3.1 Technology

Traditional motion-capture systems use a wired data logger carried in a waist harness; a multitude of wires runs from the harness to various sensors positioned on body segments of interest (typically the arms, legs, back and torso). Clearly, the use of wearable wireless sensors would greatly simplify data collection and would allow patients to wear the sensors for longer periods of time since the bulky data logger and leads would be eliminated.

Three sensor types are commonly used for motion analysis studies in the field: accelerometers, gyroscopes, and surface electrodes for electromyographic (EMG) recordings [4, 40, 5]. Triaxial accelerometers measure the orientation and movement of each body segment. Gyroscopes measure angular velocity and combined with accelerometer data can be used to accurately determine limb position [35, 20]. Surface EMG electrodes capture the electrical field generated by depolarized zones traveling along the muscle fibers during a muscle contraction. The root mean square value of the EMG data is roughly proportional to the force exerted by the monitored muscle. Thus analysis of the patterns of EMG activity can lead to the identification of motor tasks and their characteristics [3].

3.3.2 Mercury motion analysis sensor board

Our Mercury motion analysis sensor board (Figure 1(c)) interfaces to the Telos mote and incorporates a 2g/6g 3-axis accelerometer (STMicroelectronics model LIS3L02AQ), a single-axis gyroscope (Analog Devices ADXRS300) and one EMG unit (MP-1A.20.A0DM.60 from Motion Lab Systems, Inc.). The board includes a number of operational amplifiers to enhance signal quality together with voltage conditioning ICs to power the gyroscope and passive filters to eliminate noise. Signals are routed through the board to five ADC ports on the Telos mote. This is a prototype design and the next revision will include three gyroscopes mounted in a triaxial configuration.

In our proposed studies, a patient will wear several Telos motes outfitted with a Mercury board, one on each body segment of interest. In the stroke patient study, this will require one node on each of the upper arm, lower arm, back, and torso on the patient's affected side. The Telos AA battery pack will be replaced with a thin, rechargeable battery which significantly reduces size and weight.

Each axis of the accelerometers and gyroscopes is sampled at 100 Hz while the EMG is sampled at 1 kHz. Data is captured to the mote's EEPROM and relayed using a reliable communication protocol to a nearby base station for logging. Because it is necessary to correlate signals across multiple sensor devices, data from each node needs to be consistently timestamped. We are investigating various time synchronization techniques for this

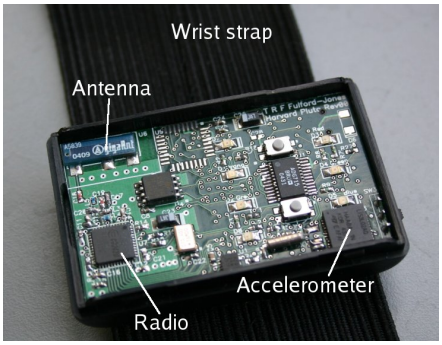


Figure 3: **The *Pluto* custom wearable mote.**

purpose [15, 39].

3.4 Pluto: A wearable wireless sensor design

The previously discussed sensor boards are intended to interface with the commercially available Mica2, MicaZ and Telos motes. While these platforms have been invaluable as a basis for experimentation, we have found that they are not ideal for wearable use in a medical setting. The use of AA batteries is convenient for testing but adds considerable size and weight (an important consideration when working with patients who suffer from motion impairment). The smaller Mica2Dot mote has been used in various settings where small size and weight are critical—however, no 802.15.4 version of this platform exists. In addition, the use of an external whip antenna is problematic in terms of packaging. Many physicians who have viewed demonstrations of our MicaZ/Telos based sensor boards have commented on their large size and apparent fragility (poor battery pack and sensor board connectors).

We have developed a custom wearable mote platform as a proof-of-concept to demonstrate tight integration of the required components in a form factor that is optimized for medical deployments. Our prototype *Pluto* mote (Figure 3) sacrifices expandability and long battery life in favor of a lightweight, miniaturized design that fits within a convenient plastic enclosure.

Pluto is based on the Telos Rev B (recently renamed "Tmote Sky") mote design [41], the schematics for which are publicly available, and incorporates the TI MSP430 microprocessor and ChipCon CC2420 radio. The board layout is about 70% of the surface area of Telos, and Pluto uses a gigaAnt surface-mount antenna instead of the inverted-F design used on the Telos. A tiny rechargeable 120 mAh lithium polymer battery powers the device. With an average current consumption of 25 mA, Pluto will run continuously for nearly 5 hours, although duty cycling to low-power modes will allow lifetimes to be considerably extended. A Mini-B USB connector is used for programming and to recharge the battery; the board features built-in recharge circuitry. No software changes are required to run TinyOS applications on Pluto, since it is 100% compatible with the Telos design.

Rather than provide expansion capabilities for external sensors, our intent is to design multiple revisions of Pluto with the required sensor components integrated onto the board; this will help minimize system dimensions for actual deployments. The first Pluto design incorporates the same STMicroelectronics 3-axis accelerometer as the Mercury motion analysis sensorboard.

Even without sensors, Pluto is useful as a "wearable tag" that can store patient information and track location using RF signals (see Section 4.5). Pluto can also be used as a rudimentary one-way communication device: the mote includes an external pushbutton that can be used by a patient to transmit an alert message to hospital staff.

Module	ROM (bytes)	RAM (bytes)
Coordinator	2140	494
CBQ	1682	244
TinyADMR	3544	1563
PulseOx	702	21
MoteTrack	5866	1035
Miscellaneous	142	24
TinyOS general	21284	181
Radio stack	7706	495
Total	43066	4057

Figure 5: **Code size breakdown for the CodeBlue software compiled for the MicaZ platform.**

Pluto is designed to fit into an inexpensive OEM plastic enclosure which measures $57 \times 36 \times 16$ mm. An elasticated wristband with velcro fasteners is attached to the enclosure. The Pluto mote, battery, enclosure and wristband weigh just 30.5g, whereas the Telos (with AA batteries and no enclosure) weighs 61g.

4 The CodeBlue Architecture

The previous section described our work on medical sensor hardware. However, supporting the diverse requirements for medical sensor networks also requires that we take a fresh look at the software environment, routing protocols, and query interfaces. In this section we describe the design and architecture for *CodeBlue*, a protocol and middleware framework for medical sensor networks. CodeBlue is implemented in TinyOS [24] and provides protocols for integrating wireless medical sensors and end-user devices such as PDAs and laptops. CodeBlue is intended to act as an "information plane" tying together a wide range of wireless devices used in medical settings.

CodeBlue is based on a *publish/subscribe routing framework*, allowing multiple sensor devices to relay data to all receivers that have registered an interest in that data. This communication model fits naturally with the needs of medical applications where a number of caregivers may be interested in sensor data from overlapping groups of patients. A *discovery protocol* is provided to allow end-user devices to determine which sensors are deployed in a CodeBlue network, while a *query interface* allows a receiving device to request data from specific sensors based on type or physical node address. The query interface also provides a filter facility, whereby a query can specify a simple predicate on sensor data that will transmit only when the data passes the filter. For example, a doctor might request data on a patient only when the vital signs fall outside of a normal range.

Figure 4 shows an overview of the CodeBlue software architecture, and Figure 5 shows the memory usage breakdown for each TinyOS component.

4.1 Publish/subscribe routing layer

As described above, CodeBlue is based on a publish/subscribe routing framework in which sensors publish relevant data to a specific *channel* and end-user devices subscribe to channels of interest. Publish/subscribe communication decouples the concerns of devices generating data from those receiving and processing it.

Any practical implementation of a publish/subscribe model must take a number of considerations into account. First, sensors should not publish data at an arbitrary rate, since the wireless channel has limited bandwidth. This implies that the communication model should either specify requested data rates or give publishers the ability to locally filter sensor data before publication. Second, given that publishers and subscribers are not necessarily within radio range, some form of multihop routing is necessary. Third, the communication layer should take mobility into account

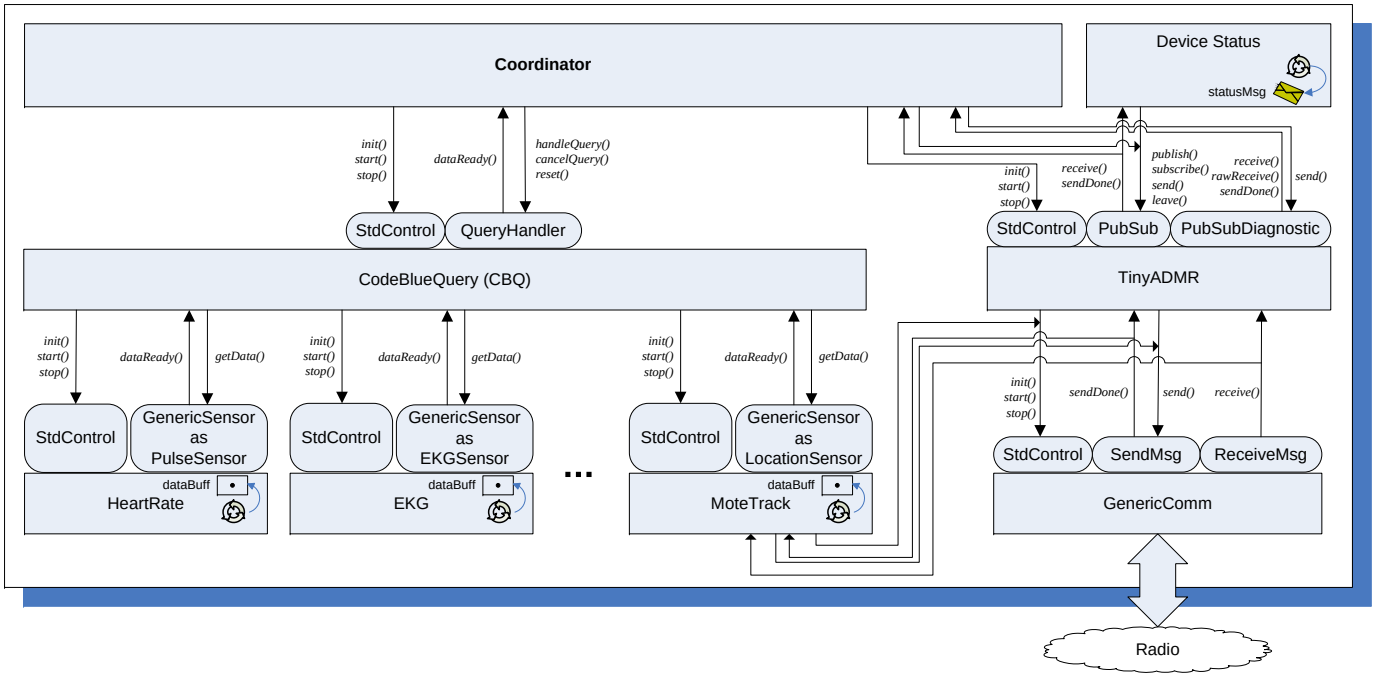


Figure 4: **The CodeBlue software architecture.**

```
interface PubSub {
  command result_t publish(uint16_t chan);
  command result_t subscribe(uint16_t chan);
  command result_t leave(uint16_t chan);

  command result_t send(uint16_t channel,
      uint8_t length, TOS_Msg* msg);
  event result_t sendDone(TOS_MsgPtr msg,
      result_t success);

  event TOS_MsgPtr receive(TOS_MsgPtr m,
      uint16_t channel, uint16_t srcAddr);
}
```

Figure 6: **The TinyADMR software interface.**

when establishing routing paths. In the medical scenario we expect both patients and caregivers to be mobile. Many patients may be ambulatory and free to roam about the hospital ward. Even those confined to hospital beds may be transferred between wards or temporarily moved for surgery or imaging.

The CodeBlue routing layer is based on the Adaptive Demand-Driven Multicast Routing (ADMR) protocol [28]. We selected ADMR because it is simple and has been extensively studied in simulation. As far as we are aware, ours is the first implementation of ADMR to be developed and tested on real hardware, and certainly the first using motes and TinyOS.

We describe the ADMR protocol only briefly here; more details can be found in [28]. The TinyADMR component provides a `PubSub` interface that exposes the commands and events shown in Figure 6. The *publish* and *subscribe* commands allow a node to state that it wishes to associate with a particular channel, while *leave* terminates a publish or subscribe request. The *send*, *sendDone* and *receive* interfaces are equivalent to their Active Message counterparts in TinyOS, except that the channel ID replaces the destination mote ID.

ADMR establishes multicast routes by assigning nodes to be *forwarders* for a particular channel. A forwarder simply rebroadcasts any messages that it receives on a given channel, using duplicate suppression to avoid multiple transmissions. Nodes are assigned as forwarders through a route discovery process that is initiated when a patient device requests to publish data. Multicast routing allows nodes to avoid transmitting redundant data; for example, if multiple doctors subscribe to vital signs from the same patient, the patient need only transmit its data once to the channel, where it will be forwarded to each recipient.

Route discovery in ADMR operates as follows. Every CodeBlue node maintains a *node table* indexed by the publisher node ID. Each node table entry contains the *path cost* from the publisher to the current node, as well as the *previous hop* in the best path from the publisher. Whenever an ADMR message is received, the node table entry corresponding to the publisher is consulted. If the estimated path cost from the publisher to the current node is lower than the node table entry (or no node table entry exists), the new previous hop and path cost fields are updated accordingly.

Routing costs can be estimated in a number of ways [10, 53]. We use an estimator of the total *path delivery ratio* (PDR) from the originating node. This estimate is based on an empirical model that maps the CC2420 radio's Link Quality Indicator (LQI) to an estimated *link delivery ratio* (LDR), using extensive measurements from our 30-node sensor network testbed. The total path loss can be calculated as $\prod_{l \in L} \mathrm{LDR}(l)$ where $\mathrm{LDR}(l)$ represents the link delivery ratio for link l (estimated from LQI of the received message), for all links L along the path from the originator to the current node. The PDR is carried in the header of each ADMR message and is updated incrementally at each hop. We have found that this path selection metric yields very reliable routes while avoiding the use of multiple rounds of message exchange to directly estimate link delivery ratios. The path cost is then (1-PDR), that is, the *path loss ratio*.

With the information in the node table, each node knows the best (lowest cost) path from each publisher to itself. When a subscriber wishes to receive data from a specific channel, it sends a unicast *route reply* message along the reverse path from itself to the publishing device, using the previous-hop information in the node table. Upon receiving the route reply, each intermediate node configures itself as a forwarder for the requested channel and will subsequently rebroadcast received messages for that channel.

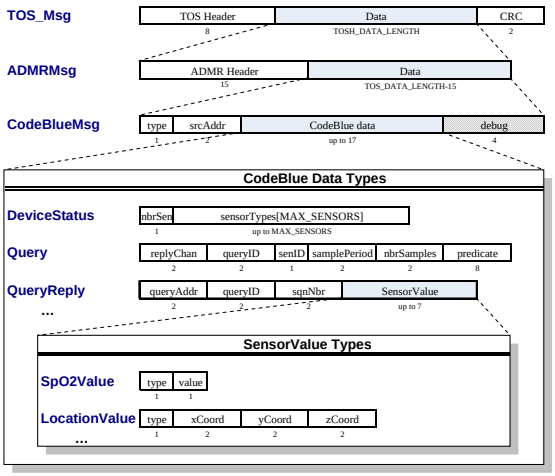


Figure 7: **CodeBlue message formats.**

Note that due to link asymmetry, the route reply message may traverse a poor link and be dropped. For this reason, we use hop-by-hop acknowledgment and retransmission *only* for relaying the route reply message to the publisher.

The route discovery process maintains the best paths from publishers to subscribers by periodically propagating a controlled broadcast flood that updates the node tables on all intermediate nodes. This periodic flooding allows the network to adapt to changes in network topology caused by node movement and environmental conditions. We currently use an update interval of 15 seconds, allowing broken routes to be repaired quickly without inducing too much protocol overhead. However, understanding the practical tradeoffs in the route management process for very large networks is worthy of further research.

4.2 Discovery protocol

In order for CodeBlue nodes to discover each other and determine the capabilities of each sensor device, a simple *discovery protocol* is layered on top of the ADMR framework. ADMR supports a special-case broadcast channel that uses a simple controlled flooding mechanism to deliver a message (unreliably) to every node in the network. Each CodeBlue node periodically publishes metadata about itself, including node ID and sensor types that it supports, to the broadcast channel. Receiving devices that wish to learn about other nodes in the network can subscribe to the broadcast channel to receive this information. Note that the metadata information about a node is static and is not updated frequently (the current update interval is 30 seconds). It would be straightforward to reduce the number of broadcast messages by performing in-network aggregation of this metadata.

4.3 CodeBlue query interface

The CodeBlue Query (CBQ) layer allows receiving devices to establish communication pathways by specifying the sensors, data rates, and optional filter conditions that should be used for data transfer. Similar to Directed Diffusion [27] and TinyDB [36], CBQ is intended to provide a very simple means of expressing data requirements in a CodeBlue network. A CBQ query is generated by an end-user device (such as a PDA or laptop) and instructs CodeBlue nodes to publish data that meets the query conditions on a specific ADMR channel.

4.3.1 Query structure

CBQ does not provide a textual interface for issuing queries; rather, queries can be issued using the GUI described in Section 4.6. A CBQ query is specified by the tuple $\langle S, \tau, chan, \rho, C, p \rangle$. S represents the set of node IDs that should report data for this query and τ is the *sensor type* representing a specific physiological sensor. Examples of sensor types include *heart rate*, *SpO_2* and *EKG*. This model allows a single node to support multiple physical sensors. Results from the query should be published to the ADMR channel *chan*. The query also specifies the sampling rate ρ and an optional count C of the total number of samples to retrieve from each node (if C is unspecified, it is assumed to be infinite). For example, the query $\langle \{3, 7\}, SpO_2, 38, 1.0\ Hz, \infty, p \rangle$ specifies that nodes 3 and 7 should report their SpO_2 data to channel 38 every second, using the filter predicate p.

The filter predicate can be used to suppress transmission of sensor data when the predicate condition is not met. It has the form

$$(\tau_1 \prec T_1)\ \text{OP}\ (\tau_2 \prec T_2)$$

where τ_1 and τ_2 are the outputs of (possibly different) sensor types and T_1 and T_2 are threshold values. $\prec$ represents a comparison operator such as $<$, $\leq$, $=$, or $\neq$. OP is one of *AND*, *OR*, or *XOR*. No more than two subexpressions can be included in the predicate; this limitation allows the predicate to fit in a single query message. For example, the query predicate (*HR* $<$ 50) *OR* (*HR* $>$ 200) would trigger data transmission only when the patient's heart rate falls below 50 bpm or exceeds 200 bpm.

Queries are currently issued to the network over the ADMR broadcast channel; this ensures that every node will receive the query, even if the set S of nodes that are responsible for processing it is small. We chose to use broadcast here because new queries are relatively infrequent events, so maintaining routing paths for query dissemination would be far more resource intensive than a rare flood. Queries are periodically re-broadcast until all patient sensors specified in the query report that they have received it. Each query has a unique ID that contains the subscriber ID, ensuring that query IDs from different subscribers do not collide. The subscriber can cancel the query with a short command (also sent to the broadcast channel) with the query ID as a parameter.

Internally, the query processor consists of two main components. The first is the *coordinator* that receives messages from the radio, handles various internal commands (e.g. for debugging), and forwards queries to the CBQ component. CBQ maintains a table of running queries, as well as a sorted queue of query execution events. Each event contains a pointer to a query as well as the time until the next event. This design allows us to use a single timer to drive the execution of all queries.

4.3.2 Discussion

CBQ's implementation is greatly simplified by the use of the underlying publish/subscribe layer. Unlike TinyDB [36] and Cougar [55], the query engine is not responsible for maintaining routing paths, nor is it concerned with how the routing topology may affect results. However, the clean separation between the CBQ and ADMR layers results in some inefficiency. For example, both CBQ and ADMR perform separate broadcast floods, the former for advertising node metadata and the latter for establishing routing paths. It is clear that a simple cross-layer optimization could be performed to combine these floods: for example, ADMR could solicit a message payload from CBQ to include in its periodic path establishment transmissions.

We believe that the basic set of predicate operators in CBQ is sufficient for most cases of interest. A more general query language, such as SQL, seems to be unnecessary for filtering medical sensor data. The simple predicate structure in CBQ allows sensor

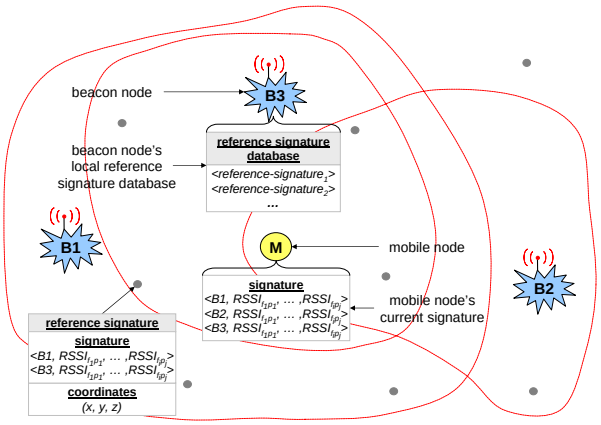


Figure 8: **The MoteTrack location system.** *$B1$, $B2$, and $B3$ are beacon nodes, which broadcast beacon messages at various frequencies and transmission powers (f_1p_1, ..., f_ip_j). Each beacon node stores a subset of all reference signatures. M is a mobile node that can hear from all three beacon nodes. It aggregates beacon messages received over some time period into a signature. The areas enclosed by perimeter lines indicate the reachability of beacon messages from the corresponding beacon node. The dots denote the known locations where reference signatures were collected.*

data from up to two separate sensors to be used to trigger transmission of a third sensor on the same patient. In addition, the set of operators exposed by CBQ can readily be expanded to include sensor-specific operations, such as detecting an arrhythmia from EKG data.

Note that CBQ includes no provisions for in-network aggregation. In general, aggregating data across multiple patients does not appear to be useful; a doctor does not want to know the *average* heart rate of all of the patients on the ward! As we gain more experience with this query model in real medical settings we expect to enhance CBQ as necessary.

4.4 Sensor interface

A `GenericSensor` interface is used to abstract the details of acquiring data from each sensor type. Like the standard TinyOS `ADC` interface, `GenericSensor` provides a simple split-phase interface. Data is requested with a call to *getData()*, which causes a *dataReady()* event to be signaled upon completion. This event returns a pointer to an internal memory buffer containing the sensor data, the size of which depends on the sensor type. Each sensor component must also provide the `StdControl` interface allowing the associated hardware to be powered on or off as necessary.

The set of sensor types supported by CBQ on a particular device is configured at compile time with a set of programmer-specified flags. These flags cause the appropriate sensor modules to be automatically wired to the CBQ component and included in the sensor metadata advertisements. In this way the binary for a sensor node will only include the components necessary for the sensors actually present.

4.5 RF-based location tracking

In many medical settings, it is extremely useful to be able to accurately locate patients, doctors, nurses, and even specialized pieces of equipment (e.g. a crash cart). For this purpose, CodeBlue incorporates a robust, decentralized RF-based localization system, called MoteTrack [34]. MoteTrack is designed to operate using only the low-power radios already incorporated into CodeBlue sensor nodes and end-user devices. In our building, MoteTrack achieves an 80th percentile location error of about 1 m, which is generally accurate enough to locate a patient or caregiver when necessary. In CodeBlue, MoteTrack is simply treated as another sensor type that reports the (x, y, z) location of the device when queried.

MoteTrack is an *empirical* localization scheme that matches the radio "signature" acquired by a roaming device with a database mapping signatures to known locations. MoteTrack improves upon systems such as RADAR [2] in that it does not require a central server to maintain the signature database; rather, this information is stored on the set of *beacon nodes* that are distributed throughout the area (e.g. a hospital). Each beacon node (which is simply a mote that may be connected to mains power) periodically transmits radio messages at a range of frequencies and transmission power levels (see Figure 8).

A mobile node listens for these beacons and acquires a signature that consists of the average received signal strength (RSSI) for each beacon node, frequency, and power level. The signature is compared to a database of pre-acquired signatures (each labeled with a known location) and a 3D location is determined. The signature database is replicated across the set of beacon nodes allowing the mapping process to be decentralized. We have explored a wide range of parameters in terms of signature distance metrics, weighting schemes, and techniques for mitigating beacon node failure; complete details are presented in [34].

Apart from enabling localization, the beacon nodes also provide a routing "backbone" for the ADMR protocol. While beacon nodes are not required by CodeBlue itself, having a fixed set of infrastructure nodes can be useful for establishing good communication coverage in an indoor environment.

4.6 User interface

The CodeBlue prototype provides a Java-based graphical user interface (GUI) that is intended to be easy for medical personnel to use and which provides enough detail on patient status and location to identify trends. We plan to work with our medical colleagues to refine the GUI for specific applications.

The CodeBlue GUI is shown in Figure 9. The upper-left panel displays a summary of metadata received from all patient sensors in the network along with the latest sensor reading. The user may request data from patient sensors by clicking in the appropriate box, which will issue a CBQ query with default parameters based on the sensor type. A more advanced interface can be used to specify CBQ parameters such as filtering predicates and data rates. Also shown in the patient list are "strength bars" indicating the network path quality to this patient sensor; this is calculated based on the path quality reported by ADMR. The user can monitor this information to ascertain whether they are experiencing undue packet loss due to the current ADMR route.

The upper-right panel shows a trace of the sensor data received from the currently-selected patient. Below this is a map of the area, showing the location of all patients for which the user has an active query. The fixed infrastructure nodes are also shown, as well as the path taken by packets routed by the ADMR protocol from the patient sensor to the end user. The message path is shown for debugging purposes only and is determined by instrumenting ADMR to include path information in each message header; this would not necessarily be turned on by default.

Each end-user device communicates with the CodeBlue network through a mote programmed with a specialized "base station" program called *PubSubBase*. Unlike the standard TOSBase code included in TinyOS, which only forwards radio messages to and from its serial port, PubSubBase understands the ADMR pro-

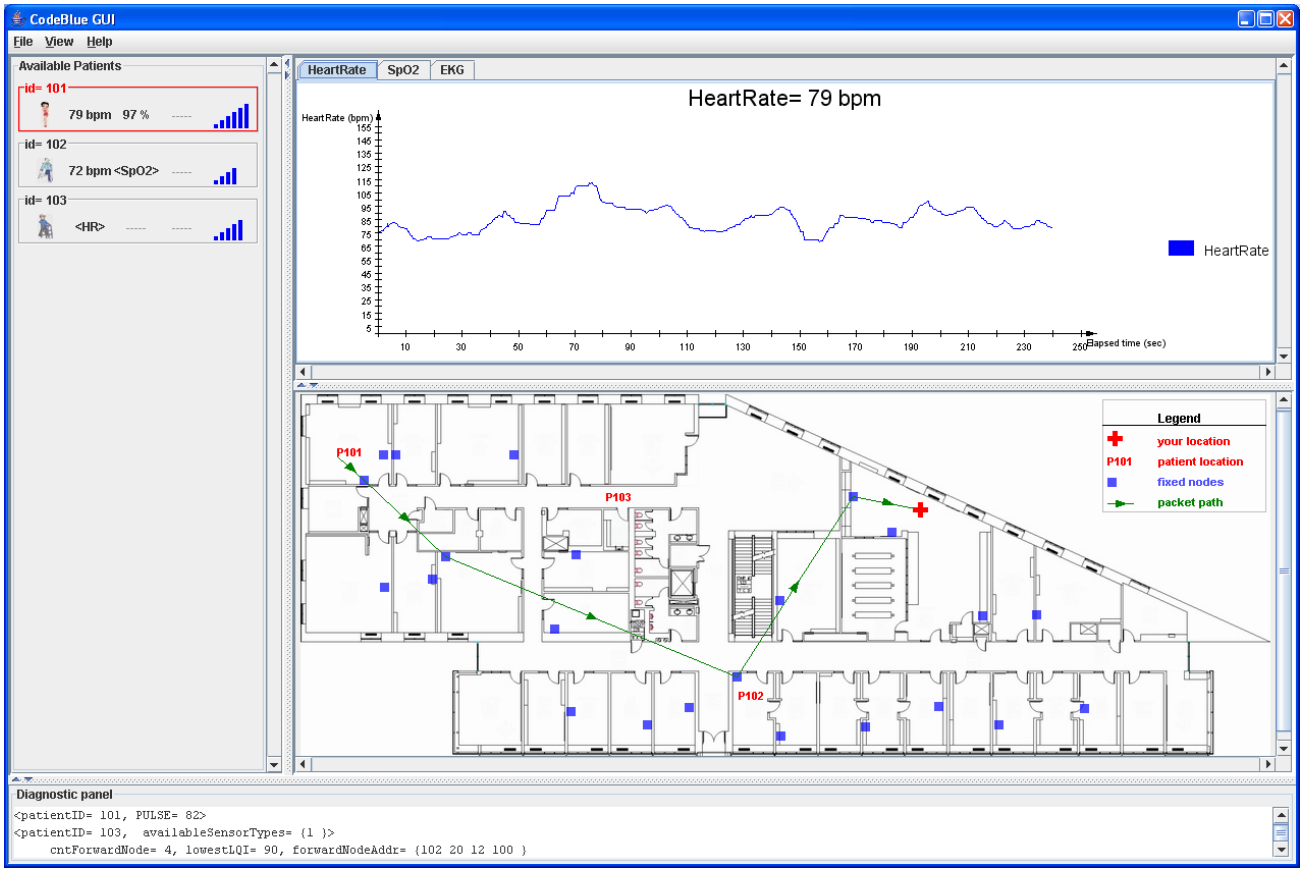


Figure 9: **The CodeBlue user interface.** *This is an actual screenshot of the CodeBlue GUI running in our building with three patient sensors reporting data to a laptop.*

tocol and provides the publish/subscribe interface to the end-user device. As a result, the complete ADMR subsystem does not need to be reimplemented on the end-user laptop or PDA.[1] This also allows us to make changes to the ADMR protocol without affecting the GUI implementation.

4.7 End-to-end use case example

To illustrate the use of the complete CodeBlue framework and GUI, we present an end-to-end use case where a doctor issues a query to a single patient sensor. The process begins when the patient's vital sign sensor (say, a pulse oximeter) is first powered on. The CBQ module uses the broadcast channel to listen for queries and to publish its metadata.

The doctor's laptop is connected to a PubSubBase mote that also listens on the broadcast channel. It receives the patient sensor metadata, unwraps the message payload (containing the patient node ID and sensor types), and passes the information to the Java GUI over its serial port. The GUI then displays the ID and sensor types for the new patient in the patient list panel (see patient 103 in Figure 9). The doctor can issue a query for the patient's vital signs by double-clicking on the icon in this panel, or may elect to issue a more complex query with filtering parameters.

The query message is passed to the PubSubBase where it is forwarded on the broadcast channel until it reaches the patient sensor. The CBQ module on the patient device interprets the query message and passes it to the query processor module for execution. The query processor samples the user's pulse oximeter at the specified rate, interprets the filtering predicate (if any) and relays the query results to the CBQ module. CBQ then transmits the vital sign data on the destination channel specified by the user's query. ADMR routes this data to the PubSubBase connected to the doctor's laptop, which in turn relays the data to the Java GUI for display.

[1] We are developing a variant of the Telos mote with a Compact Flash interface that will provide direct radio connectivity to PDA-class devices.

5 Evaluation

In this section we present an initial evaluation of the CodeBlue system running on an indoor testbed of 30 MicaZ motes, distributed over 3 floors of our Computer Science building. Although the location of each node is fixed, this testbed affords us the opportunity to measure communication reliability and throughput under a wide range of link conditions and data rates. We also present results demonstrating the use of CodeBlue with mobile receivers.

Our goal in evaluating CodeBlue is to validate its overall robustness and scalability with multiple transmitting and receiving devices. We also wish to explore the effect of increased data rates on achieved throughput. Our results are promising and show that CodeBlue and ADMR achieve good packet delivery ratios with modest data rates. However, radio bandwidth saturation is a serious problem with higher data rates, suggesting that this should be a primary focus for future work.

5.1 Evaluation environment

Our 30-node sensor network testbed provides a Web-based interface allowing users to schedule time and run jobs on the testbed. The system also forwards messages to and from each mote's serial port via a TCP socket, allowing us to control and monitor the entire network from a single machine. We have implemented a Java-based driver to send commands to the CodeBlue nodes for

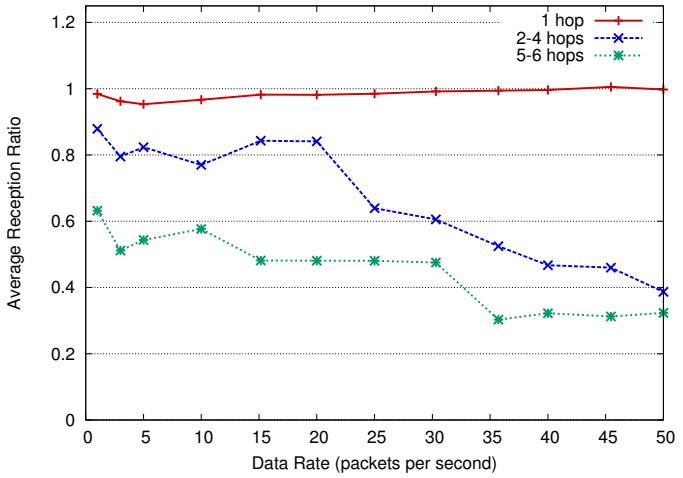


Figure 10: **The effect of increasing data rate and hop count on reception ratio.** *This experiment measures three separate sender-receiver pairs with different number of radio hops in the ADMR path. Increasing the transmission rate leads to degradation in reception rate due to dropped packets.*

issuing queries, receiving data, retrieving statistics, and so forth. This setup proved to be very convenient, making it possible to run tests with many different parameters without having to reprogram the motes each time.

In each experiment, we used "virtual" sensors on each patient device that generate data at a constant rate. Each experiment was executed for at least 2 minutes, and statistics were calculated after removing the first 60 seconds of each trace to avoid measuring startup effects. Of course, this does not directly measure latency for query propagation and route establishment. Our results do indicate that a doctor or nurse coming onto a shift will be able to issue queries and receive results with a lag time of *at most* 1 minute.

5.2 Scalability

The first set of experiments attempts to measure three scalability-related properties of our system:

- What is the effect of increasing the data rate generated by each sensor device?
- What is the effect of increasing the number of senders?
- What is the effect of increasing the number of receivers?

Because of the limited size of our testbed, we are unable to directly generate data for very large networks (hundreds of nodes or more). However, we can emulate this behavior by increasing the data rate from each transmitter, which increases background traffic.

Varying data rate and hop count: Figure 10 shows the packet reception ratio (the number of received packets divided by the number of transmitted packets) for three separate sender-receiver pairs. In all three cases, the same node is used as the sender, while the receiving node is varied. Receivers were selected to vary the number of radio hops along the ADMR path. Note that the hop count varies over time because ADMR routes are dynamic. The single-hop case should be very common in clinical settings where the doctor or nurse is generally near the patient.

As the figure shows, the reception rate is very good in the single-hop case, even for high data rates (50 packets per second). With the multi-hop cases, the reception ratio degrades substantially. This occurs for two reasons. First, in multi-hop cases, forwarding nodes must compete for bandwidth with both the upstream and downstream forwarders, limiting the amount of available bandwidth for each node. Increased reception rates cause packet queues on each node to fill, eventually forcing packets to be dropped. Second, we have observed that increasing the amount of interfering traffic adversely affects reception ratios on our testbed, even when no transmissions are dropped. This is likely due to collisions caused by hidden-terminal effects.

Varying number of senders: Apart from varying the data rate for each sender, we can explore the effect of varying the number of senders. In each case we use the same receiving node but increase the number of senders from 1 to 10. In each case the *per-sender* data rate is increased from 1 to 50 packets per second. In the single-sender case the receiver is within radio range, but in other cases the senders are between 1 and 4 hops away. It is worth noting that as we added senders, the average hop count increased as well, so we would expect throughput to degrade more seriously in those cases.

The results shown in Figure 11 are encouraging: for low data rates (below 5 packets per second per sender), the reception ratio is above 62%, even with 10 senders. Given that many vital sign sensors (pulse oximetry, blood pressure, heart rate) only need to transmit data at most once a second, this suggests that the system could scale to a large number of devices each with a modest data generation rate.

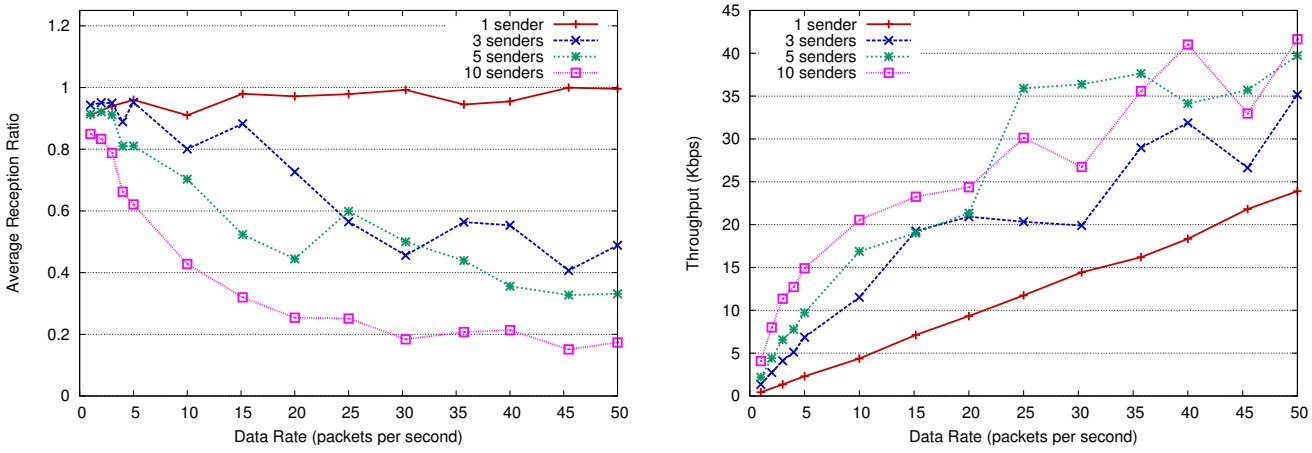
Varying number of receivers: We repeated these experiments with 3 separate receivers, using the same set of senders; the results are shown in Figure 12. Because the receivers are no longer within one hop from the first sender, even in the one-sender case we see some degradation as the data rate increases. Figure 12(b) shows that the maximum aggregate bandwidth of 10 senders and 3 receivers is 120 Kbps, or 40 Kbps per receiver. This is consistent with tests that we have performed measuring the single-hop throughput of Telos motes with the standard TinyOS CC2420 radio stack. These numbers are far below the nominal 802.15.4 channel capacity of 250 Kbps due to MAC and protocol overheads.

It is worth noting that the reception ratio degradation with 3 receivers is much less than what we would expect if constructing *unicast* paths between senders and receivers. To get a rough idea of what the latter case would entail, consider the reception ratio for 10 senders and 1 receiver in Figure 11(a). If the sender is generating data at a rate of 10 packets per second and relaying data to 3 receivers via unicast, this is roughly equivalent to the node generating data at 30 packets per second, which results in a reception ratio of about 20%. However, the multicast case (Figure 12(a)) shows that with 10 senders, 3 receivers, and a per-sender data rate of 10 packets per second, we achieve a reception ratio closer to 40%. This shows that multicast routing in CodeBlue helps to mitigate the effects of bandwidth limitations.

5.3 Fairness

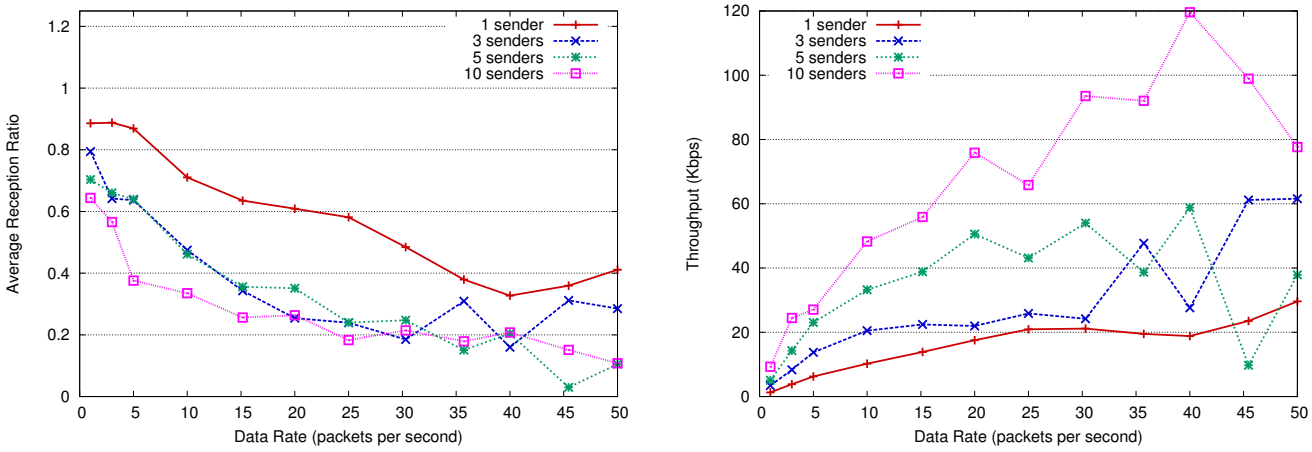
In a multicast environment with multiple publishers and subscribers, we are concerned with the overall *fairness* achieved by the routing substrate. If the network unfairly biases certain paths over others, a doctor receiving data from the system has little confidence that the data they are receiving is evenly distributed across patient sensors.

To demonstrate the overall fairness of the CodeBlue routing layer we ran an experiment with 6 senders (each generating data at 1 packets per second) and 3 receivers. The path hop counts for each route varied from 1 to 6. Figure 13 shows the reception ratio



(a) Reception ratio **(b) Aggregate throughput**

Figure 11: **Effect of increasing data rate and number of senders with 1 receiver.** *Reception ratios are very high for data rates below 5 packets per second, even with 10 separate senders over multihop paths (1-5 hops).*



(a) Reception ratio **(b) Aggregate throughput**

Figure 12: **Effect of increasing data rate and number of senders with 3 receivers.** *Increasing the number of receiving nodes has a more serious effect on the reception ratio as data rates are increased. (Path hop counts in this test ranged from 1 to 7).*

breakdown across each sender-receiver pair. As the figure shows, the reception ratio across all pairs is roughly equivalent, with a mean of 83% and a standard deviation of 12%. In only one case was the reception ratio less than 60%.

5.4 Latency and jitter

The scalability results show that bandwidth limitations are a serious issue for delivery of medical data in CodeBlue. Apart from reduced packet reception ratios, we are concerned about the potential impact on packet *latency* induced by background traffic. In addition, we are interested in studying the pattern of packet loss; that is, whether losses occur in large bursts or more intermittently.

Latency: Measuring packet latency in a multihop network is difficult and would require either fine-grained synchronization between senders and receivers or a round-trip measurement. Time synchronization using a protocol such as FTSP [39] is possible, although the results would be dependent on FTSP's own accuracy in our testbed. Round trip measurements are problematic in the ADMR framework because different paths would be chosen in the forward and reverse directions.

Instead, we chose to instrument the message path in CodeBlue by having senders, receivers, and forwarders send debug messages to their serial ports during the routing process. These messages are received by the central testbed server and timestamped with an accuracy of a few milliseconds; however, because this time stamping involves several context switches on the (loaded) server it is unclear how accurate this is.

We measured the end-to-end latency for several multihop paths of up to 7 hops at a data rate of 1 packet per second. The end-to-end message delay was measured to be less than 200 ms in all cases. Through link-level measurements in our testbed, we have measured the MAC delay of the TinyOS radio stack under a wide range of traffic conditions. The delay varies between 3 ms (with no background traffic) to about 15 ms (with heavy background traffic). We believe this range generally characterizes expected per-hop packet latencies in the CodeBlue environment.

Packet jitter: We define *packet jitter* as the number of consecutive dropped packets for a given sender-receiver pair. This can be measured by comparing packet sequence numbers on the receiver. If the jitter were very large, we would be concerned that much critical medical data would be lost. Figure 14 shows a histogram for a single sender-receiver pair placed at an average distance of 5 hops. As the figure shows, the packet reception ratio is about 70%. In 22% of the cases, the jitter was equal to 1; in less than 8% of the cases the jitter is 2 or more packets. In no case was a jitter of more than 5 packets observed.

To understand how multiple senders and receivers affect packet jitter, we repeated the previous experiment with 6 senders and

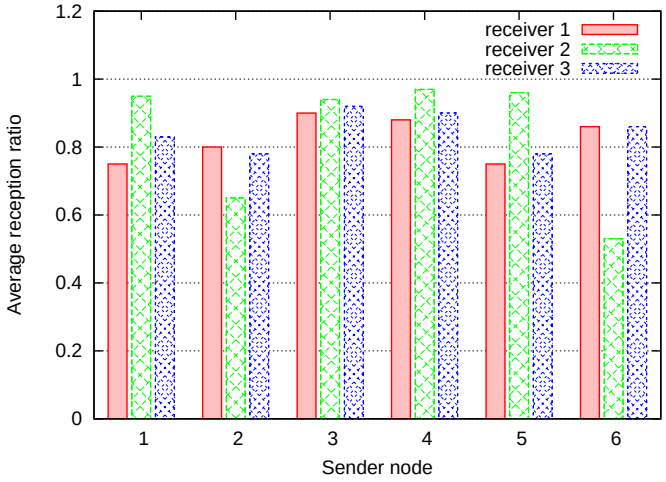


Figure 13: **Fairness across multiple sender-receiver pairs.** *This graph shows the reception ratio breakdown across 18 sender-receiver pairs with a data rate of 1 packets per second.*

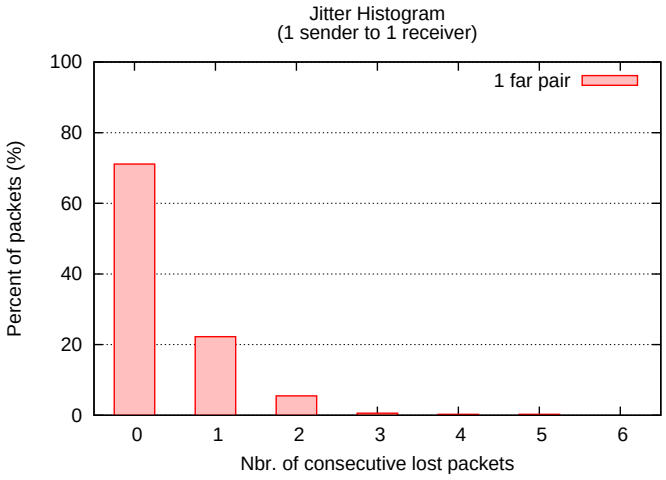


Figure 14: **Packet jitter distribution for a single node pair.** *The sender and receiver are placed at opposite ends of the building with an average path hop count of 5.*

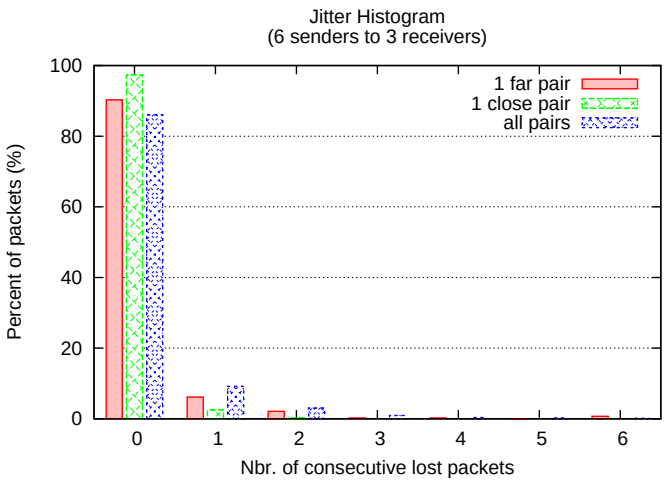


Figure 15: **Packet jitter distribution across 6 senders and 3 receivers.** *This graph shows packet jitter for 3 cases: a single-hop node pair, a multi-hop pair, and across all 18 node pairs.*

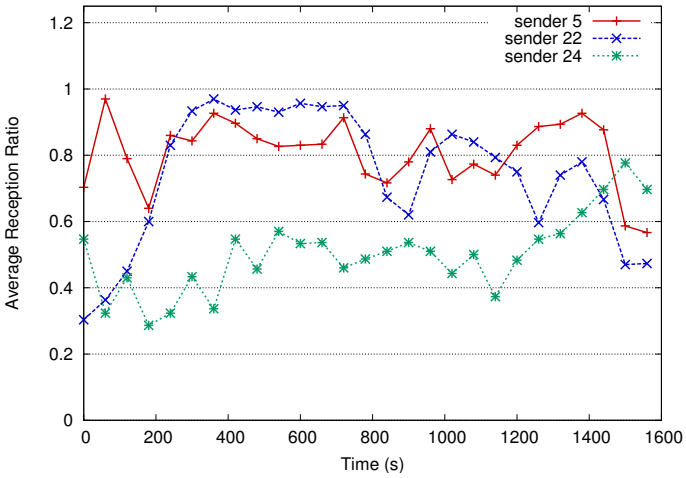


Figure 16: **Effect of mobility.** *Reception ratio averaged over 60 second intervals for 3 senders and a single roaming receiver.*

3 receivers distributed throughout the building, transmitting one packet per second. Figure 15 shows the jitter histogram for all 18 pairs. The figure also shows the jitter for two specific paths: the multihop pair from our previous experiment with 1 sender and 1 receiver, and a single-hop pair. No jitter is observed for 86% of the packets, 9% of packets experienced a jitter of 1, and 5% of the packets experienced a jitter greater than 1. The maximum jitter observed in this case is 23 packets. It is interesting to note that with multiple senders and receivers, jitter is reduced for the first pair of nodes (the same pair that was measured in Figure 14). This is explained by the increased number of forwarders, which increases the chance of packets getting through.

5.5 Effect of mobility

The final evaluation that we wish to present concerns the impact of mobility on communication reliability. As senders or receivers move in a hospital, radio link quality will vary and ADMR will create new routes. Therefore, we expect to see some data loss due to node mobility, but ideally a valid route will be maintained at all times.

In this experiment, we configured 3 fixed nodes as patient sensors transmitting data at 5 packets per second. The senders were widely distributed throughout the building. A single receiver node attached to a laptop acted as a roaming node. The user carrying the laptop moved around the second floor of our building at a normal walking pace, pausing occasionally, entering and leaving rooms, for a duration of about 25 minutes. This movement pattern is intended to represent a doctor walking through a hospital ward.

Figure 16 shows the reception ratio for each of the 3 senders, averaged over 60 second windows. As the receiver walks around, we see the reception ratios vary over time, but do not see any large dropouts or catastrophic effects due to mobility. We have also recorded the hop count and ADMR path cost for each packet and see a general correlation between improved delivery ratio and reduced path cost. These results show that ADMR deals gracefully with node movement, at least for "typical" mobility rates.

5.6 Mitigating packet loss

Although CodeBlue does not currently provide a reliable routing mechanism, we anticipate that reliable communication will be necessary for many medical scenarios, especially clinical studies and continuous monitoring during surgery. Thus far, our focus has been on unreliable multicast which allows the system to scale to many patient sensors and receiving devices. We expect that medical sensor networks will require a range of reliability semantics

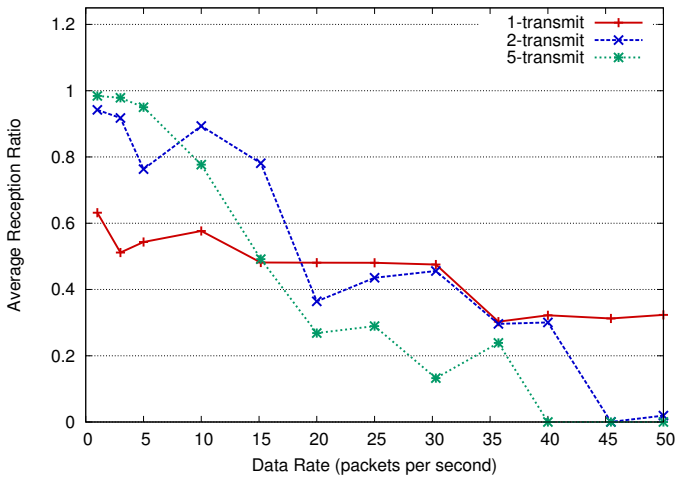


Figure 17: **Mitigating packet loss by transmitting each message multiple times.** *Data is shown for a single multihop path.*

for different types of data.

The best approach to implementing reliability is not immediately clear. Using link-by-link acknowledgment and retransmission with multicast requires additional MAC support and may incur high overhead. End-to-end reliability is highly sensitive to overall path conditions.

One approach that is worth considering makes use of redundant transmissions and coding techniques that allow data to be reconstructed on the receiver despite packet loss. We are still investigating this idea, but to capture a rough estimate of how it would perform we have conducted experiments where each message is simply transmitted multiple times by the sender. In this way a receiver can recover the original data if any one of k transmitted packets is received. This approach consumes considerably more bandwidth but should yield an estimate of the improvement obtainable via more sophisticated techniques.

Figure 17 shows the result of a series of experiments with a single multihop path (with an average path length of 5 hops). As the figure shows, for low data rates (below 15 packets per second), using multiple transmissions per packet increases robustness considerably, from 63% (with 1 transmission) to over 98% (with 5 transmissions). However, at larger data rates the increased message load causes network saturation and reception ratios drop considerably. Ideally, nodes would be able to tune their transmission rates according to background traffic conditions.

6 Future Work and Conclusions

Our evaluation of the CodeBlue prototype points to a number of critical areas for future work. The most serious is the lack of reliable communication, although our results show that this problem can be mitigated somewhat through redundant transmissions. We do not believe that reliable routing is required for all medical data; rather, the system should allow each query to specify its reliability needs in terms of acceptable loss, data rate, or jitter.

Another area worth exploring is the impact of bandwidth limitations and effective techniques for sharing bandwidth across patient sensors. For example, each CodeBlue query could specify a data priority that would allow certain messages (say, an alert from a critical patient) to have higher priority than others in the presence of radio congestion. This approach can be combined with rate-limiting congestion control [14, 25] to bound the bandwidth usage of patient sensors.

An important shortcoming of the current CodeBlue prototype is its lack of security. We have already begun to explore the integration of private-key encryption [29] along with a public-key protocol for key distribution [38, 22] in CodeBlue. The privacy and security requirements for medical care are complex and differ depending on the scenario. For example, HIPAA privacy regulations need not be enforced during life-saving procedures. Nevertheless, we intend to integrate some form of end-to-end security into the next version of the CodeBlue system.

In conclusion, this paper has presented an initial exploration into the challenges of hardware and software design for medical sensor networks. We believe that low-power wireless sensors have the potential for tremendous impact in many medical applications. We have described a range of mote-based medical sensors as well as a prototype protocol and middleware platform. CodeBlue is currently being developed through several active collaborations with local hospitals and we anticipate a clinical deployment of the system in the next few months.

References

[1] A & D Medical, Inc. UA-767BT Wireless Blood Pressure Monitor. `http://www.lifesourceonline.com/products/telemonitoring.cfm`.

[2] P. Bahl and V. N. Padmanabhan. RADAR: An in-building RF-based user location and tracking system. In *Proc. INFOCOM (2)*, pages 775–784, 2000.

[3] P. Bonato, R. Hughes, D. Sherrill, R. Black-Schaffer, M. Akay, B. Knorr, and J. Stein. Using Wearable Sensors to Assess Quality of Movement After Stroke. In *Proceedings of the 65th Annual Assembly American Academy of Physical Medicine and Rehabilitation 2004*, Phoenix, Arizona, October 2004.

[4] P. Bonato, P. Mork, D. Sherrill, and R. Westgaard. Data mining of motor patterns recorded with wearable technology. *IEEE Eng Med Biol Mag.*, 22(3), May-June 2003.

[5] J. Bussmann, J. Tulen, E. van Herel, and H. Stam. Quantification of physical activities by means of ambulatory accelerometry: a validation study. *Psychophysiology*, 35(5), 1998.

[6] Card Guard. Wireless Blood Pressure Monitors. `http://www.cardguard.com/site/products-list.asp?id=20`.

[7] A. Cerpa, J. Elson, D. Estrin, L. Girod, M. Hamilton, and J. Zhao. Habitat monitoring: Application driver for wireless communications technology. In *Proc. the Workshop on Data Communications in Latin America and the Caribbean*, Apr. 2001.

[8] Crossbow Technology, Inc. MICA2 Series (MPR4x0). `http://www.xbow.com/Products/productsdetails.aspx?sid=72`.

[9] Crossbow Technology, Inc. MICAz ZigBee Series (MPR2400). `http://www.xbow.com/Products/productsdetails.aspx?sid=101`.

[10] D. S. J. De Couto, D. Aguayo, J. Bicket, and R. Morris. A high-throughput path metric for multi-hop wireless routing. In *Proceedings of the 9th ACM International Conference on Mobile Computing and Networking (MobiCom '03)*, San Diego, California, September 2003.

[11] J. Deng, R. Han, and S. Mishra. A performance evaluation of intrusion-tolerant routing in wireless sensor networks. In *Proc. Second International Conference on Information Processing in Sensor Networks (IPSN'03)*, April 2003.

[12] E. Dishman. Inventing wellness systems for aging in place. *IEEE Computer*, 37(5), May 2004.

[13] Dolphin Medical. OEM 601 Oximetry Module. `http://www.dolphinmedical.com/oem601.htm`.

[14] C. T. Ee and R. Bajcsy. Congestion control and fairness for many-to-one routing in sensor networks. In *Proc. SenSys'04*, Baltimore, MD, November 2004.

[15] J. Elson, L. Girod, and D. Estrin. Fine-grained network time synchronization using reference broadcasts. In *Proc. Fifth Symposium on Operating Systems Design and Implementation (OSDI 2002)*, Boston, MA, December 2002.

[16] Federal Communications Commission. Wireless Medical Telemetry Service (WMTS). `http://wireless.fcc.gov/services/personal/medtelemetry/`.

[17] T. R. F. Fulford-Jones, G.-Y. Wei, and M. Welsh. A portable, low-power, wireless two-lead EKG system. In *Proc. 26th IEEE EMBS Annual International Conference*, San Francisco, September 2004.

[18] GE Healthcare. ApexPro CH. `http://www.gehealthcare.com/usen/patient_mon_sys/wireless_and_telemetry%/products/telemetry_sys/products/apexpro_ch.html`.

[19] GE Healthcare. Corometrics 340M – Telemetry Ambulatory monitoring during labor. http://www.gehealthcare.com/usen/perinatal/mat_fetal_mon/products/colo3%40M.html.

[20] D. Giansanti, V. Macellari, G. Maccioni, and A. Cappozzo. Is It Feasible to Reconstruct Body Segment 3-D Position and Orientation Using Accelerometric Data? *IEEE Trans Biomed Eng*, 50(4), 2003.

[21] GMP Wireless Medicine, Inc. LifeSync Wireless EKG System. http://www.wirelessecg.com/.

[22] N. Gura, A. Patel, A. Wander, et al. Comparing elliptic curve cryptography and RSA on 8-bit CPUs. In *Proc. Cryptographic Hardware and Embedded Systems (CHES 2004): 6th International Workshop*, Cambridge, MA, August 2004.

[23] HealthFrontier, Inc. ecgAnywhere. http://www.healthfrontier.com.

[24] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. E. Culler, and K. S. J. Pister. System architecture directions for networked sensors. In *Proc. the 9th International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 93–104, Boston, MA, USA, Nov. 2000.

[25] B. Hull, K. Jamieson, and H. Balakrishnan. Techniques for mitigating congestion in sensor networks. In *Proc. SenSys'04*, Baltimore, MD, November 2004.

[26] IEEE. IEEE 802.15.4 — Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs), October 2003.

[27] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *Proc. International Conference on Mobile Computing and Networking*, Aug. 2000.

[28] J. G. Jetcheva and D. B. Johnson. Adaptive Demand-Driven Multicast Routing in Multi-Hop Wireless Ad Hoc Networks. In *2001 ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc2001)*, pages 33–44, October 2001.

[29] C. Karlof, N. Sastry, and D. Wagner. TinySec: A link layer security architecture for wireless sensor networks. In *Proc. Second ACM Conference on Embedded Networked Sensor Systems (SenSys 2004)*, November 2004.

[30] D. Konstantas, V. Jones, R. Bults, and R. Herzog. Mobihealth - innovative 2.5/3g mobile services and applications for healthcare. In *Proc. Eleventh IST Mobile and Wireless Telecommunications Summit 2002*, Thessaloniki, Greece, June 2002.

[31] K. V. Laerhoven, B. P. Lo, J. W. Ng, S. Thiemjarus, R. King, S. Kwan, H.-W. Gellersen, M. Sloman, O. Wells, P. Needham, N. Peters, A. Darzi, C. Toumazou, and G.-Z. Yang. Medical healthcare monitoring with wearable and implantable sensors. In *Proc. Sixth International Conference on Ubiquitous Computing*, Tokyo, Japan, September 2004.

[32] L. Lenert et al. WiiSARD: Wireless Internet Information System for Medical Response in Disasters. https://wiisard.org.

[33] B. Lo and G. Z. Yang. Key technical challenges and current implementations of body sensor networks. In *Proc. 2nd International Workshop on Body Sensor Networks (BSN 2005)*, April 2005.

[34] K. Lorincz and M. Welsh. MoteTrack: A Robust, Decentralized Approach to RF-Based Location Tracking. In *Proceedings of the International Workshop on Location- and Context-Awareness (LoCA 2005) at Pervasive 2005*, Oberpfaffenhofen, Germany, May 2005.

[35] H. Luinge, P. Veltink, and C. Baten. Estimating Orientation with Gyroscopes and Accelerometers. *Technol Health Care*, 7(6), 1999.

[36] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. TAG: A Tiny AGgregation Service for Ad-Hoc Sensor Networks. In *Proc. the 5th OSDI*, December 2002.

[37] D. Malan, T. Fulford-Jones, M. Welsh, and S. Moulton. CodeBlue: An ad hoc sensor network infrastructure for emergency medical care. In *Proc. MobiSys 2004 Workshop on Applications of Mobile Embedded Systems (WAMES 2004)*, June 2004.

[38] D. Malan, M. Welsh, and M. Smith. A public-key infrastructure for key distribution in TinyOS based on elliptic curve cryptography. In *Proc. the First IEEE International Conference on Sensor and Ad hoc Communications and Networks (SECON)*, Santa Clara, CA, October 2004.

[39] M. Maroti, B. Kusy, G. Simon, and A. Ledeczi. The flooding time synchronization protocol. In *Proc. SenSys'04*, Baltimore, MD, November 2004.

[40] M. Mathie, A. Coster, N. Lovell, and B. Celler. Accelerometry: providing an integrated, practical method for long-term, ambulatory monitoring of human movement. *Physiol Meas.*, 25(2), April 2004.

[41] Moteiv, Inc. Tmote sky.

[42] Nonin Medical, Inc. Avant 4000 Wireless Wearable Pulse Oximeter. http://www.nonin.com/products/4000.asp.

[43] L. Ohno-Machado et al. SMART: Scalable Medical Alert Response Technology. http://smart.csail.mit.edu/.

[44] J. Polastre, R. Szewczyk, and D. Culler. Telos: Enabling ultra-low power wireless research. In *Proc. Fourth International Conference on Information Processing in Sensor Networks: Special track on Platform Tools and Design Methods for Network Embedded Sensors (IPSN/SPOTS)*, April 2005.

[45] T. Rusch, R. Sankar, and J. Scharf. Signal processing methods for pulse oximetry. *Computers in Biology and Medicine*, 26(2):143–159, 1996.

[46] E. Shih, V. Bychkovsky, D. Curtis, and J. Guttag. Demo abstract: Continuous, remote medical monitoring. In *Proc. Second Annual International Conference on Embedded Networked Sensor Systems*, November 2004.

[47] Smiths Medical PM, Inc. BCI Micro Power Oximeter Board. http://www.smiths-bci.com/html/Products/oem_products.htm.

[48] R. Szewczyk, A. Mainwaring, J. Polastre, and D. Culler. An analysis of a large scale habitat monitoring application. In *Proc. Second ACM Conference on Embedded Networked Sensor Systems (SenSys)*, November 2004.

[49] K. K. Tremper and S. J. Barker. Pulse oximetry. *Anesthesiology*, 70(1):98–108, January 1989.

[50] Welch Allyn, Inc. Micropaq Wireless Patient Monitor. http://www.monitoring.welchallyn.com/products/wireless/micropaq.asp.

[51] G. Werner-Allen, J. Johnson, M. Ruiz, J. Lees, and M. Welsh. Monitoring volcanic eruptions with a wireless sensor network. In *Proc. Second European Workshop on Wireless Sensor Networks (EWSN'05)*, January 2005.

[52] D. White et al. AID-N: Advanced Health and Disaster Aid Network. https://secwww.jhuapl.edu/aidn/.

[53] A. Woo, T. Tong, and D. Culler. Taming the underlying challenges of reliable multihop routing in sensor networks. In *Proc. the First ACM Conference on Embedded Networked Sensor Systems (SenSys 2003)*, November 2003.

[54] G.-Z. Yang et al. Body Sensor Network Node. http://www.doc.ic.ac.uk/vip/ubimon/bsn_node/index.html.

[55] Y. Yao and J. E. Gehrke. The Cougar approach to in-network query processing in sensor networks. *ACM Sigmod Record*, 31(3), September 2002.