Proceedings of the 28th IEEE
EMBS Annual International Conference
New York City, USA, Aug 30-Sept 3, 2006

Design and Implementation of a Framework for Monitoring Patients in Hospitals Using Wireless Sensors in Ad Hoc Configuration

Nicholas O'Donoughue, *Student Member, IEEE*, Sarvesh Kulkarni, *Member, IEEE,* Douglas Marzella

***Abstract*—Patients in hospital Intensive Care Units (ICUs) have to be monitored constantly. Typically, unless the attending physician is at the bedside, he or she has no information of the patient's progress, and must be paged manually in case of emergencies. A far better system would be one that keeps the patient's information available to the doctor or nurse at all times, possibly through the use of a handheld device. The doctor would then be able to check on the patient's progress in real time. Should an emergency arise, the doctor would be notified directly by the monitoring system itself, saving valuable response time.**

This paper aims to set up an infrastructure for monitoring patients in hospital ICUs. An architectural framework and related protocols to support communication between the various components of such a system are presented. The prototyped product was successfully demonstrated to industry representatives, in November 2005, at Villanova University, PA.

Keywords—**ad hoc sensor networks, telemedicine, real time data acquisition, remote monitoring**

I. INTRODUCTION

Hospital Intensive Care Units (ICUs) are extremely busy. Doctors and nurses are constantly being called to attend to patients in need of urgent care. When an emergency arises, the nurse must first recognize the danger and then page a doctor, who must then rush to the patient's aid. Thus, precious time is wasted in responding to the emergency. Another problem is that doctors do not have easy access to patient information; they must be at a monitoring station or at the patient's bedside in order to see how a patient is progressing. Even in non-emergency situations, the doctor and the patient must often be in the same place before care can be administered. As a consequence, there has been much research directed towards applications in telemedicine.

A. State of Current Research

Numerous proposals have been made recently for systems that use fixed infrastructure to bring patients and physicians in separate locations together for diagnosis and monitoring [5, 7, 9, 12]. These systems focus either on a specific application, such as transmission of ECG data through the WWW [9], or the general case of monitoring unspecified parameters at one [5, 12] or multiple [7] homes. These systems aim to provide additional support for out-patients at high risk of medical complications.

The advent of low-cost, wireless devices has also spawned several implementations for data collection using personal networks [1, 2, 6, 13] and long-range wireless communications [5, 8, 13]. Bhargava et al. [1] discuss some of the security considerations necessary for a medical ad hoc network. In response to developments of proprietary and vendor-specific implementations, Varady et al. [15] propose an open architecture for bedside networks that, while inherently wired, could easily be extended to the wireless realm of devices.

In addition, advances in ubiquitous computing have encouraged applications for displaying data on mobile computing devices, such as PDAs and Pocket PCs [4, 11].

Thus, significant attention has been paid to the areas of telemedicine and automated data acquisition. The proposed and implemented solutions aim to provide remote healthcare in the home and on the move. What has been missing from all of these systems, however, is the integration of mobile data acquisition *with* mobile data retrieval.

B. Our Proposed System

The intelligent system that we propose seeks to alleviate these shortcomings. Patient data will be collected at the bedside much like it currently is. However, we propose to send the collected data through a wireless ad hoc network to an Ethernet Gateway Node (EGN). The EGN will then log this data to a database using a program called the Data Acquisition and Alert Module (DAAM) running on a server. The DAAM checks each patient's vital signs against a pre-defined safe range as they are received. The DAAM transmits this information over the Internet to be received by a doctor or a nurse carrying a Palm OS hand-held device such as a Personal Digital Assistant (PDA). A program, the 'ICUClient,' installed on the PDA receives automatically generated alerts from the DAAM whenever a patient slips into a critical condition. It also allows a doctor to proactively request and view specific patient data graphically in real time.

Currently a wireless sensor network consisting of commercially available Crossbow Mica2 and Mica2Dot motes is being used to mimic patient data (see www.xbow.com). The sensors collect data regarding such things as temperature, sound, and light intensity, which we treat as relevant patient data, for testing purposes. The proposed system has been fully designed, developed and

Manuscript received 12 July 2006.

Nicholas O'Donoughue , Sarvesh Kulkarni and Douglas Marzella are with the Department of Electrical and Computer Engineering at Villanova University, Villanova, PA 19085 USA (phone: 610-519-4970, fax: 610-519-4436; e-mail: {nicholas.odonoughue, sarvesh.kulkarni, douglas.marzella} @villanova.edu).

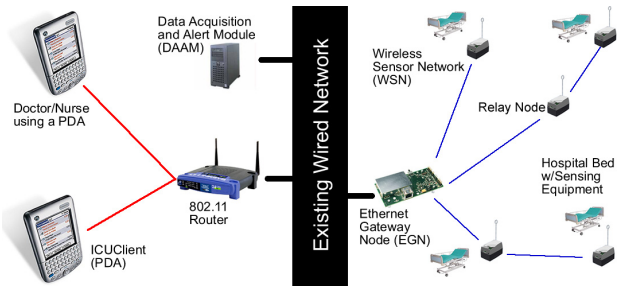

Fig. 1. Overview of patient information retrieval system. The WSN (right) consists of several nodes connected wirelessly to an EGN. The DAAM (top center) is loaded onto a PC. The ICUClient is a software program that resides on PalmOS handhelds (left) with IEEE 802.11.

tested (although not in a hospital environment). A demonstration was conducted at the annual Villanova University ECE Day on November 3rd, 2005.

Section II contains an overview of the design, subdivided into the three main system components. Section III discusses the design process. Section IV explains the current state of implementation. Section V concludes the paper.

II. Design Overview

The design of our system consists of three main components, as seen in figure 1. The first subsystem, the wireless sensor network (WSN) is responsible for collecting and transmitting all relevant patient data. The second subsystem, the DAAM, is responsible for recording and monitoring all received data, as well as managing connections with all of the PDA clients. The third subsystem, the 'ICUClient', is responsible for accessing the DAAM to retrieve patient data, and for displaying that data on the user's PDA. The expected users of this system are the doctors and nurses working in a hospital ICU.

A. Wireless Sensor Network (WSN)

The WSN consists of Crossbow Mica2 and Mica2Dot motes, which operate in the unlicensed 900 MHz band. The motes are pre-configured with the 'Surge Reliable Route' routing algorithm, which is available from Crossbow, Inc. The base station of the WSN, the Crossbow MIB 600, receives all of the data transmissions from the motes routed over one or more wireless hops and makes them available over the attached Ethernet port. A "node" is any mote in the WSN. A node may have multiple sensors attached to it, or none at all. The nodes form a cooperative, multi-hop, ad hoc wireless network. In our proposed solution, patients would be issued a unique node (with sensors attached). That node would collect all sensor data for that particular patient and transmit it for storage to the database in the DAAM.

B. Data Acquisition and Alert Module (DAAM)

The second subsystem is the DAAM. A graphical view is provided in figure 2. The DAAM consists of several modules and a MySQL database. The main component in the DAAM is the Logger module, which opens a socket to the EGN and stores all of the data into a MySQL database. The Logger is also responsible for monitoring the incoming data in real time for what we have described as "emergency conditions," *i.e.* those which signify that a patient is in need of immediate attention. Whenever an incoming data point is flagged as an emergency, an alert is sent via a method call to the 'PalmServer' module (described below).

The MySQL database consists of three tables. The first table in the database, "nodes," is a list of all nodes in the network, which contains pertinent information about each node, including time of last transmission. The second table, called "packetReceived," is a list of each transmission that was received from the sensor network, as well as all the data carried in those transmissions. The third table, "safeRange," is a list of the accepted safe ranges for each data set.

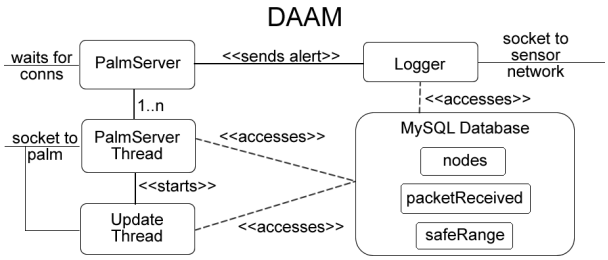

Fig. 2. Block diagram of DAAM.

The 'PalmServer' module is responsible for accepting incoming connections from users, and creating an instance of the 'PalmServerThread' module for each connection. The PalmServer also distributes all alerts that are sent by the Logger, so that each user can be notified as appropriate.

The PalmServerThread module handles communications with the ICUClient, queries the MySQL database in response to requests for data, and formats the results appropriately before transmission. This module is used to send information to the user about what nodes are "active" in the sensor network, and to provide a "snapshot" of the most recent data transmission received from a selected node. If the user wishes to receive streaming information on a specific data set (such as pulse or body temperature), then the PalmServerThread hands this task off to a new instance of the 'UpdateThread' module. The UpdateThread module queries the MySQL database for all transmissions received from the specified node within recent history, and then transmits the data points, in order, to the user. This frees up the PalmServerThread to listen for messages from the user without any lag in response time.

C. ICUClient

The 'ICUClient' software runs on the Palm PDA. All logic control for this program lies in its 'ICUClientMIDlet' class, with all socket functionality handled by the 'Communicator' class. The five 'screen' classes comprise the visual interface, and the Graph class handles the actual plotting of received data streams.

The user can browse active nodes, view 'snapshots' of the most recent transmission from a node, and view streaming data from a specific node and sensor graphically. While the user is browsing, the system is always ready to receive and

immediately display an alert if one is sent by the DAAM.

D. Communications Protocol

A necessary aspect of our system is the communications protocol that we designed. We devised eight different message types to facilitate communication between the DAAM and the ICUClient. All of these message types are subordinates to the parent *PalmMsg* class. Table 1 has a brief description of all of the message types.

TABLE I
MESSAGE TYPES

Type	Purpose
PalmMsg	Parent class.
NodeListMsg	Send list of active nodes to ICUClient.
SnapshotMsg	Send node snapshot to ICUClient.
DataMsg	Send streaming data point to ICUClient.
AlertMsg	Send node alert to ICUClient.
RefreshMsg	Request update of node list from DAAM.
NodeRequestMsg	Request node snapshot from DAAM.
SensorRequestMsg	Request streaming data from DAAM.

III. DESIGN PROCESS

A. Wireless Sensor Network

The first phase of implementation consisted of setting up and troubleshooting our WSN test bed. Eight Mica2 motes (four of which were paired with the MTS310 sensor boards) and four Mica2Dot motes (two of which had MTS510 sensor boards) were configured into a sensor network. An MIB600 Ethernet gateway (with one of the eight Mica2 motes attached) was configured as a bridge between the wireless and wired portions of the network. We began by installing the Cygwin and TinyOS environments onto a development machine. Next, we loaded the Surge Reliable Route routing algorithm onto the motes, and deployed those motes in an indoor testing environment. We used the vendor-provided applications to graphically visualize the network and ensure that all nodes were operating correctly. We then placed nodes in different arrangements and locations to test the ad hoc capacity of the network, and reduced the transmission power to force the nodes to route over multiple hops, to ensure that the network was behaving as we expected.

B. Data Acquisition and Alert Module

The DAAM was loaded onto a Dell Optiplex GX240 with a 1.70 GHz Pentium 4 processor and 512MB RAM, running Windows XP. MySQL Server 4.1 was also loaded onto the Dell to run the database, and TinyOS 1.1.13 with Cygwin was installed to program the motes and run the DAAM.

We began by gathering raw transmissions from the nodes and deciphering the contents using the approach in [14]. Then, we interfaced the Logger with the MySQL database, and verified that packets were being appropriately archived in the database as entries in the packetReceived table.

During the first iteration of emergency handling, the safe ranges were hard coded into the Logger, until we could verify that our logic was correct. Then, we created a table in the MySQL database to contain these values, and rewrote the Logger software to utilize the safeRange table.

The PalmServer module itself is straightforward, consisting of only three main parts. The first is a socket server, which receives incoming socket connections and reallocates each one to an open port. The second part is an array of PalmServerThread processes, one of which is instantiated with each new connection, and the third part is an alert function, which is called by the Logger module whenever the users need to be alerted of an emergency.

The PalmServerThread module begins by querying the MySQL database for a list of all the currently active nodes (defined as those that have sent a transmission in the last five minutes), which is sent to the user. Then, the PalmServerThread waits for incoming requests and responds appropriately. If streaming data is required, the PalmServerThread creates an instance of the UpdateThread module to execute the request. The PalmServerThread is also ready to transmit an alert to the ICUClient immediately upon notification of an emergency.

The final module is the UpdateThread, the sole purpose of which is to stream data to the user. Upon instantiation, the thread queries the database for all recent packets transmitted from the desired node, and then transmits the data points, in order, to the user. This process is repeated indefinitely, until the parent PalmServerThread terminates it.

C. ICUClient: Development Environment Selection

Before choosing a development environment, we researched three different applications: CodeWarrior, Onboard C, and Palm OS Developer Suite (PODS). We chose to begin with PODS for reasons of cost. After several failed attempts to use the C communication libraries to open a socket, we discovered Java 2 Micro Edition (J2ME), which greatly simplifies software development on the Palm OS platform. Detailed instructions on the development of J2ME MIDlets for Palm OS can be found in [10].

D. ICUClient: Component Description

The ICUClient is made of eight classes. This module was designed extensively and implemented one class at a time, with thorough incremental testing. The first class, ICUClientMIDlet class controls the logic flow of the program. Next is the Communicator, which is responsible for handling all communications with the server. The WelcomeScreen, NodeListScreen, SnapshotScreen, IPConfigScreen, and DataScreen classes are all used for display. The Graph class is used exclusively by the DataScreen in order to display real time information in a graphical format for the end user.

The WelcomeScreen is the first screen seen by the user upon starting the ICUClient application. Once at the WelcomeScreen, the user has two options: connect, or configure the connection settings for the DAAM. The second option takes the user to the IPConfigScreen, which allows the user to change the IP address and port number of the desired host server, and ensures that the entries are valid.

After a connection is established, the NodeListScreen displays the nodes that are actively connected and allows the

user to select one, in order to see more detailed information about that node. The SnapshotScreen (figure 3, left) displays more detailed information about a sensor once it is selected. It displays the most recent values that have been received from all of the sensors attached to a node. From this list, a specific sensor can be selected to request streaming data.

The final aspect of this implementation is the receipt and display of streaming data from the server. We first created the DataScreen object, an extension of the J2ME Canvas class, to display the text. The DataScreen utilizes the Graph class to generate and display visual plots (figure 3, right).

The ICUClient software was loaded onto a Palm Tungsten C, which runs Palm OS 5.2 and has a built-in IEEE 802.11b radio.

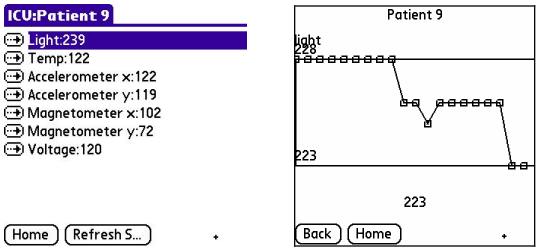

Fig. 3. ICUClient screenshot showing a snapshot of node 9's sensors (left) and streaming data from a sensor attached to node 9 (right).

IV. Current State of Implementation

At present, the prototype hospital ICU patient monitoring system is fully functional and works as designed. The system has been tested using data generated asynchronously by environmental sensors (in lieu of actual patient monitoring sensors) in real time. The sensor data is collected and relayed over an ad hoc network with multi-hop wireless links. The system generates alerts when the sensor data falls outside the determined "safe-range", and routes the alerts correctly to the Palm PDA. Sensor data can be viewed on the PDA as long as it is within range of an accessible wireless network.

A concern of note is that the wireless channels used by the nodes are susceptible to RF interference. A suitable physical layer needs to be selected to make these channels more resistant to interference-related failures. An important aspect of our design is the independence of the proposed framework (and related communication protocols) from the physical (radio) layer, thus increasing its versatility.

Finally, security-related aspects concerning the gathering and transportation of patient data must be addressed to ensure compliance with HIPPA regulations [3], and are a topic for future work.

V. Conclusion

In this paper, we present a broad framework for the automated monitoring of patients in a hospital ICU environment. The proposed framework has a straightforward implementation and does not require the development of any specialized hardware, and is thus expected to be cost-effective. A proof-of-concept system was also designed and implemented using off-the-shelf components. The system has the potential to reduce the response time of physicians to medical emergencies in hospital ICUs. It does not rely on any specific physical (radio) layer, and is thus open for use with a wide variety of frequencies and modulation schemes such as FDMA, TDMA, OFDM, etc. The only remaining hurdles that we anticipate are (a) the physical interfacing of existing hospital sensing equipment with the radio nodes of the ad-hoc sensor network, and (b) the transmission and storage of patient information in a manner that complies with HIPAA regulations.

Acknowledgment

The authors wish to thank Dr. Elliot Sloane (Villanova University) for his enthusiastic support and guidance.

References

[1] Anu Bhargava, Mike Zoltowiski, "Sensor and Wireless Communications for Medical Care," in *Proc. 14th Int'l Workshop on Database and Expert Systems Applications (DEXA '03)*, pp.956-960, 2003.

[2] Tia Gao, Dan Greenspan et. al., "Vital Signs Monitoring and Patient Tracking Over a Wireless Network," in *Proc. 27th Annual Int'l Conf. of the IEEE EMBS*, Shanghai, September 2005.

[3] HIPAA Basics: Medical Privacy in the Electronic Age, December 2005, Privacy Rights Clearinghouse, 19 February 2006, <http://www.privacyrights.org/fs/fs8a-hipaa.htm>.

[4] Kevin Hung, Yuan-Ting Zhang, "Implementation of a WAP-Based Telemedicine System for Patient Monitoring," *IEEE Trans. Information Technology in Biomedicine*, Vol. 7, No. 2, pp.101-107, 2003.

[5] KY Kong, CY Ng, K Ong, "Web-Based Monitoring of Real-Time ECG Data," *Computers in Cardiology*, Vol. 27, pp.189-192, 2000.

[6] Srdjan Krco, Vlado Delic, "Personal Wireless Sensor Network for Mobile Health Care Monitoring," in Proc. *6th Int'l Conf. on Telecommunications in Modern Satellite, Cable and Broadcasting Service*, Serbia and Montenegro, Nis, 1-3 Oct 2003.

[7] Ho Sung Lee, Seung Hun Park, Eung Je Woo, "Remote Patient Monitoring Service through World-WideWeb," in *Proc. 19th Int'l Conf. of IEEE/EMBS*, pp.928-931, 1997.

[8] Yuan-Hsiang Lin et. al, "A Wireless PDA-Based Physiological Monitoring System for Patient Transport," *IEEE Trans. Information Technology in Biomedicine*, Vol. 8, No. 4, pp.439-447, 2004.

[9] Farah Magrabi, Nigel H. Lovell, Branko G. Celler, "A web-based approach for electrocardiogram monitoring in the home," *Int'l Journal of Medical Informatics*, Vol. 54, pp.145-153, 1999.

[10] Mahmoud, Qusay, "MIDP for Palm OS 1.0: Developing Java Applications for Palm OS Devices," Sun Developer Network, January 2002, <developers.sun.com/techtopics/mobility/midp/articles/palm/index.html>.

[11] SP Nelwan et. al., "Ubiquitous Mobile Access to Real-time Patient Monitoring Data," *Computers in Cardiology*, Vol. 29, pp.557-560, 2002.

[12] Seung-Hun Park et. al., "Real-Time Monitoring of Patients on Remote Sites," in *Proc.20th Annual Int'l Conf. of IEEE/EMBS*, pp.1321-1325, 1998.

[13] C.S. Pattichis et. al., "Wireless Telemedicine Systems: An Overview," *IEEE Antennas and Propagation Magazine,* Vol. 44, No. 2, pp.143-153, April 2002.

[14] Thorn, Jeff, "Deciphering TinyOS Serial Packets," Octave Tech Brief #5-01, 10 March 2005, <http://www.octavetech.com/pubs/TB5-01%20Deciphering%20TinyOS%20Serial%20Packets.pdf>.

[15] Peter Varady et. al., "An Open Architecture Patient Monitoring System Using Standard Technologies," *IEEE Trans. Information Technology in Biomedicine*, Vol. 6, No. 1, pp.95-98, 2002.