HIP: Health Integration Platform

Jonathan Woodbridge Hyduke Noshadi Ani Nahapetian Majid Sarrafzadeh
UCLA Computer Science Department
Los Angeles, California, USA
{*jwoodbri,hyduke,ani,majid*}*@cs.ucla.edu*

***Abstract*—This paper introduces a new software development platform specifically designed for wireless health applications. Wireless health applications follow a unique paradigm encompassing body sensor networks. These networks are controlled by a central processing unit (such as a cell phone or PDA) that provides connectivity to health care professionals over a wide area network. With such architectures, health monitoring becomes ubiquitous allowing patients a greater degree of freedom from conventional medical monitoring.**

This paper introduces HIP - a wireless health integration platform. HIP is a complete end to end software development platform. Wireless Health platforms previous to HIP have two main faults. First, these platforms lack generality and focus on few wireless health components. Second, they offer an extremely rigid platform that is difficult to integrate into preexisting research. HIP addresses these issues by providing a flexible and modularized plug and play architecture that seamlessly integrates into preexisting work.

***Keywords*-Wireless Health; Software Platforms; Pervasive Health**

I. Introduction

Wireless health applications are often implemented as a Wireless Body Area Network (WBAN)[1][6][7]. Such architectures consist of a series of sensors attached to vital areas throughout the body. These sensors are typically attached to sensor motes where data is propagated to a central processing unit, such as a cell phone or PDA. Central processing units store, process and/or transmit data to a central server. Central servers perform computationally intensive analysis on data and archive it for future analysis. Architectures typically consist of several heterogeneous pieces of hardware making development and integration extremely difficult. This is often exacerbated by stringent requirements put forth by life critical applications. HIP was developed to meet these needs.

Figure 1 shows an example of wireless health architecture. The figure contains a WBAN that interfaces with several embedded systems both on and off the body. Connectivity between embedded systems is accomplished over Bluetooth while WiFi and GPRS accomplishes long range connectivity to health care professionals.

This publication was partially supported by Grant Number T15 LM07356 from the NIH/National Library of Medicine Medical Informatics Training Program

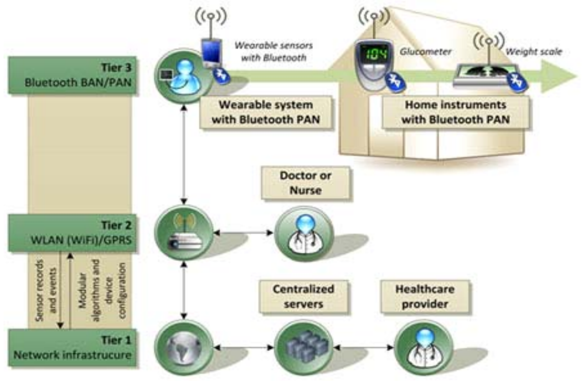
Tier 3
Bluetooth BAN/PAN
Wearable sensors with Bluetooth
Glucometer
Weight scale
104
Wearable system with Bluetooth PAN
Home instruments with Bluetooth PAN
Doctor or Nurse
Tier 2
WLAN (WiFi)/GPRS
Sensor records and events
Modular algorithms and device configuration
Centralized servers
Healthcare provider
Tier 1
Network infrastrucure

Figure 1. An example wireless health architecture. Applications consist of embedded sensor motes, a central controlling unit (for coordination of embedded sensors), and a central server for data aggregation and analysis by health care professionals.

Software development and simulation frameworks exist in virtually all well-established fields in computer science. Computer networks have simulation frameworks such as NS2[12] and Qualnet[14] and compilers have frameworks such as Mach-SUIF[9]. These frameworks assist researchers in their development of new and exciting prototypes while adding the ability to test and assess their feasibility. Such frameworks add tremendous value in time, code quality, and standardized comparison methods. HIP was developed to meet the needs of an emerging wireless health field. HIP addresses three main issues:

1) Provide a generic development framework for wireless health applications based on a WBAN architecture.
2) Provide a testing and analysis framework to test correctness and performance of wireless health applications.
3) Provide a standardized framework for analysis and comparison of research.

HIP is a complete software platform developed in Java with the ability to run on mobile phones, PDAs, PCs and any other platform that supports Java's CLDC 1.0 requirements (including J2ME and J2SE). With the introduction of Squawk and the SunSPOT embedded platform[15], HIP also runs on sensor motes. HIP is a complete end-to-end wireless health framework that runs on all components. Our framework expedites development, ensures high reliability and robustness, and provides an extensive analysis and testing framework. HIP is currently used on several research projects. In section V we discuss several ongoing projects that leverage HIP.

HIP offers a suite of features with the ability to run at all levels of a wireless health application, from a sensor mote to a central server. HIP provides an extensive analysis framework that collects a holistic set of metrics. Metrics can be collected in real time or with archived or simulated data. HIP is offered as a completely open source framework.

II. Related Work

Other work has presented a need for a wireless health platform. Authors in [8] discussed a need for interoperability and standards to produce robust and reliable wireless health platforms. These authors presented UbiMon: a platform for continuous monitoring of physiological data. UbiMon moves towards a system for supporting multiple pieces of hardware, such as the Berkley Mote and TI MSP430. However, the underlying system still depends on TinyOS. HIP was designed to be independent of sensor motes. While HIP provides libraries that run on Sun's Squawk architecture, HIP also provides a modular binding system that allows for integration of any embedded system. In general, HIP is focused on the central processing unit (such as a PDA, PC, or mobile phone). The sensor mote environment is extremely heterogenous by necessity. Different applications require different pieces of hardware. With this notion in mind, HIP provides a binding system that includes any type of sensor mote. SPINE is the most similar platform to HIP [5]. SPINE is similar to HIP in several aspects. First, SPINE is built around a similar architecture; Wireless Body Area Networks (WBAN). This architecture includes a central processing unit that collects data from sensors located throughout the body. SPINE also includes source code for both the central processing unit as well as sensor motes. This software includes several modules such as data collection, signal processing and data transfer. These libraries are extensible to fit the needs of a plethora of wireless health applications.

While analyzing SPINE, we found two major limitations. First, SPINE is built upon a rigid architecture that only supports sensor motes with Tiny OS (such as the Berkeley Motes). While these motes are impressive pieces of hardware, we can not expect all wireless health applications to only use the Berkeley Mote. Second, SPINE does not offer a complete testing and simulation environment. Its often difficult to test health applications without the use of live subjects. As researchers, we must limit the amount of pain and suffering inflicted on subjects. For this reason, a wireless health software framework must include exhaustive test and simulation features thereby limiting the use of live subjects.

CodeBlue is a wireless health platform built for disaster response [10]. In CodeBlue, each patient is connected to one to several sensor motes that observe health analytics (such has EKGs and blood pressure). These sensor motes create an ad-hoc network for relaying data to health care professionals. Health care professionals track and prioritize patients through a mobile device such as a PDA or mobile phone. CodeBlue provides the facilities for ad-hoc network formation, routing, network aggregation of data and security. HIP was designed for health applications based on the WBAN model which have a different set of requirements than that of a disaster response systems. WBAN's do not require complex ad-hoc networks and are largely dependent on wide area networks for communications with health care professionals and back-end processing engines.

Authors in [16] discuss the topic of mobile health applications and services termed m-health. They've produced MobiHealth to facilitate the production of m-health services upon next generation (2.5g/3g) mobile networks. These applications are concentrated on a WBAN architecture and require constant connectivuty. Our work is quite complementary to MobiHealth. HIP provides a modular plug and play middleware in which applications and services can be built upon. Support for short range and long range communications are inherent to the design.

III. HIP

The most recent version of our software platform is fully functional and is currently used in several research projects in our labs. HIP was designed with several requirements in mind stemming from both academia and industry. The design of HIP is based on the following three tenets:

1) Be a catalyst for fast prototyping and development.
2) Provide a complete testing and analysis framework
3) Provide a standardize method for analyzing new research.

HIP is implemented in Java and requires CLDC 1.0. Our platform is compatible with both J2SE and J2ME. By choosing Java, our platform is compatible with all major PC operating systems such as Linux, Windows, and MAC OSX. HIP is also compatible with all mobile platforms supporting J2ME (or J2SE) including Symbian OS, Windows Mobile OS, and Blackberry OS as well as Sun's Squawk [15] (a CLDC 1.1 compliant VM optimized for resourced constrained embedded platforms). A more detailed justification for our choice of Java can be found in [18].

HIP is far more than just a wireless health software platform. As tenet three states, HIP is intended to increase collaboration between researchers and industry. By providing a standardized platform, developers can easily share implementations as well as pool raw data. We see HIP as an opportunity to reach out to the wireless health community providing a means to consolidate knowledge from around the world.

It is important to note that HIP does not intend to provide a standardized data format. Work to standardize data formats and data communication was done in [3][4]. However, the current implementation does provide one data format. Developers may convert old data to a supported form, or they can create new modules to support their format. Of course, these new modules can be shared or even rolled into HIP's core software.

HIP provides a common framework for comparing scholarly work. Without a common framework such as HIP, it is often difficult to effectively discern differences between similar results. Also, implementations are not always accessible and may be implemented in completely different environments. Networking solved these issues with tools such as Qualnet and NS2. When using these tools, developers can focus their development on their respective protocols while reusing all other pieces of the network stack. Statistical tools wrap these modules providing a standardized and reproducible method for comparison. HIP follows the same methodology. Developers can replace modules with their own implementations, run their modules on a set of pre-existing (or live) data, and gather various statistics to compare with previous work. All aspects of HIP can be modified in this fashion, from data storage and compression algorithms to data processing and filtering algorithms.

A. Architecture

Our first revision of HIP consists of six main modules: data acquisition, data storage, data processing, data transfer, module analysis and event logging. These modules are discussed in detail in section IV. HIP contains complete implementations of all six modules including sample applications. Our platform was designed to work out of the box. As HIP matures, new implementations will be added to the core framework allowing developers to mix and match modules to fit their needs.

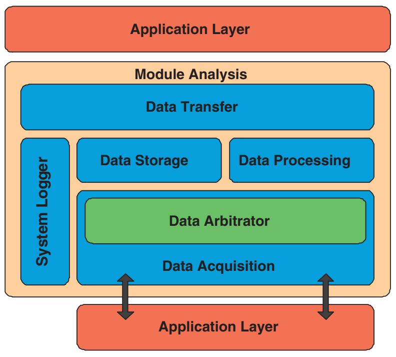

Figure 2. HIP's Overall Architecture. HIP is encapsulated by a module analysis engine for assessing all points of a wireless health application. HIP also provides integration points for both an application layer, motes, and sensors (including those accessed wirelessly).

B. Note on Sensor Motes

This paper focuses largely on the SunSPOT embedded platform. However, SunSPOT may not fit the needs of every wireless health project, For that reason, HIP was designed to interact with any embedded platform. We currently interface with the MicroLEAP[1], Texas Instruments EZ430 [2], and SunSPOT embedded platforms.

IV. Module Design

The following section describes the details of HIP's 6 main modules. HIP provides a predefined interface for each module as well as default implementations.

A. Data Acquisition

Data acquisition modules are an abstraction layer between the framework and raw data. Raw data generally is obtained from sensors or raw data stored on the local files system. However, HIP makes no restrictions on the origins of data. Data can be streamed from any source including a network, Bluetooth, Zigbee, serial, or file connection. When running HIP on an embedded processor (such as the SunSPOT), on board sensors are accessed through this API. We can also send preformatted data over a wireless protocol (such as Zigbee) from a PC to the embedded processor for testing and performance analysis.

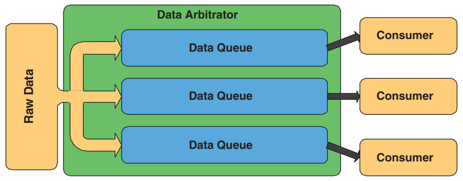

Figure 3. The data arbitrator splits raw data into a seperate stream for each consuming object. Data streams are completely independent from their siblings thereby ensuring consistent data to each consumer.

Data sources may have multiple consumers. To allow sharing of data, HIP provides a data arbitrator that wraps a stream of data. Consumers register themselves with a data source thereby creating a dedicated stream allowing a class to consume data at their own rate while avoiding contention. Under high memory demand, data may be dropped from a slow consumer's queue. However, a consumer may define it's dropping behavior to better fits its needs.

B. Data Transfer

Data transfer includes many aspects of wireless health applications. Data may be transferred as one large binary or streamed in real time. Data may travel from a sensor mote to a mobile phone or PC. The mobile phone or PC may in turn forward this info to a central server where data is analyzed and archived. A PC or mobile phone may also stream archived data from a central server to simulate real time data. HIP is designed to account for all such scenarios. In fact, our framework makes very few assumptions on how data is transferred providing extensible data transfer functionalities.

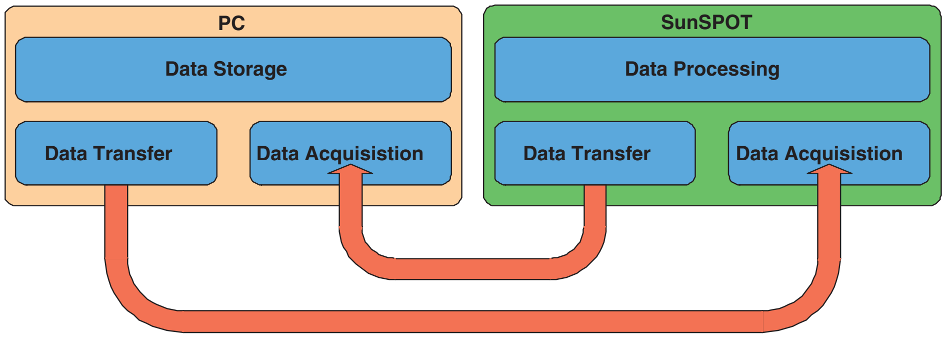

Figure 4. Simulation of sensor data is sent from a laptop to the SunSPOT embedded platform. This allows us to simulate real-world and corner case scenarios using pre-fetched data. Such simulation can be used to test performance and system integration.

In highly constrained environments such as the SunSPOT and other embedded devices, network channels are limited. The SunSPOT, for instance, only supports one open data stream in each direction. To account for such constraints, the data transfer module allows several potential senders to share an output stream. The data transfer module takes care of all multiplexing and passes the data as one stream. The receiver's data acquisition module is responsible for demultiplexing this data and passing it to the proper destinations.

C. Data Storage

There are many aspects to data storage, such as file size and access speeds. The Data Storage abstraction layer hides implementation details from the calling classes. This allows data storage implementations to be swapped at both compile time and at run time with minimal modifications to the rest of the system. This ability allows HIP's analysis engine to effectively compare different data storage implementations.

HIP also allows the ability for exposing stored data as sensor data. This ability comes with many features of an actual live sensor such as dynamic changes of sampling frequencies. This feature of HIP allows for controlled module and system testing on predefined data. Systems may also use this functionality to replay data. This feature works on both sensor motes (such as Sun's SunSPOT) as well as mobile devices and PCs.

D. Data Processing

Data processing modules are designed for both pre and post processing of gathered data. Such processing may consist of compression/decomprecion, data transformations (such as the Fourier or Gabor transforms), security, as well as techniques for compressed sensing. Of course, this is just a small subset of features facilitated by the data processing modules. HIP currently provides compression/decompression modules. However, we plan to release several more data processing libraries as HIP matures.

The data processing modules affords two main features: code reuse and a generic means of comparison. By providing a structured interface, researchers and developers can easily plug in their own custom modules and share them with the entire wireless health community. By utilizing a common platform such as HIP, comparisons of new work to past work becomes highly accurate thus improving the quality of work and respective results.

V. Experiences with HIP

We outline three ongoing projects that use HIP including SmartShoe (which demonstrates HIP's ability to integrate into a pre-existing project), analysis of a compression method, and Spinal Bridge.

A. SmartShoe

SmartShoe is an ongoing research project in our labs. One of SmartShoe many capabilities is to detect fall risk in patients. SmartShoe contains a a 3-axis gyroscope, 3-axis accelerometer, and several pressure sensors embedded within the sole of the shoe. A MicroLEAP [1] processor is used to collect and wirelessly transmit SmartShoe' data over a Bluetooth connection. Raw data is collected by a central processing unit (such as a cell phone or PDA) where data is analyzed. Data is later propagated to a central server for archival and further analysis.

SmartShoe's architecture is representative of many wireless health applications. There are two sensor motes (one for each shoe) that send data wirelessly to a central processing unit. As SmartShoe progressed, it required a robust central processing unit that collected, analyzed, and transfered data to a central server.

We chose to integrate SmartSoe with HIP for three reasons. First, we required a modular plug and play architecture that allows us to integrate new functionalities seamlessly. Second, SmartShoe required the need to collect, store, and replay data. With such functionality, we can use data collected during medical trials to test future algorithms. Finally, HIP's analysis modules allows us to analyze performance of new algorithms. This gives us an effective means to compare our work with others. Analysis features combined with the simulation framework yields an effective understanding of our implementation before ever going to field trials.

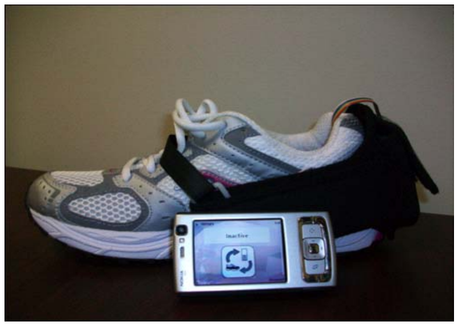

Figure 5. The complete wearable SmartShoe system. The Nokia n95 is pictures here running a mobile SmartShoe application built upon HIP's framework. Gait parameters are calculated in real time using a custom HIP module. HIP facilities communication between embedded processors within the shoe as well as health care professionals via WiFi, GSM, and text messaging services.

Figure 5 displays the wearable portion of SmartShoe. The mobile device (Nokia N95) runs balance detection software built upon HIP's framework. The shoe used the MicroLEAP embedded system that transmits data over bluetooth. The MicroLeap does not run HIP. However, we could replace the MicroLEAP with a SunSPOT for a complete end to end solution built upon HIP.

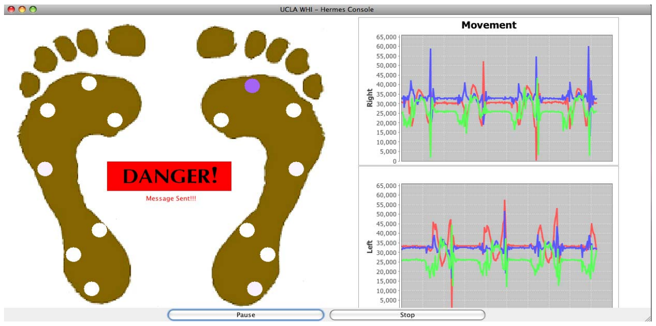

Figure 6. SmartShoe application for detecting imbalance built using HIP's framework. The application reads, stores and displays both live and past data. Gait parameters are extracted from SmartShoe in real time to analyze a patient's locomotion.

Figure 6 displays a PC application built upon HIP's framework. This application gathers and calculates the a patients fall risk. HIP libraries are identical for both the PC and mobile devices displayed earlier. This includes custom modules for determining a patient's fall risk. The portability of HIP affords a robust and consistent environment across all aspects of a wireless health application.

B. Data Compression

Data compression is extremely important when dealing with low powered embedded systems. Transmitting and receiving over a radio accounts for approximately 60% of the total energy expenditure[13]. However, analyzing the effectiveness of compression algorithms on live data is difficult. Typically, raw data is collected then compressed later to analyze the effectiveness of an algorithm. However, this process does not allow us to determine the correctness or performance of the algorithm from the sensor mote. HIP makes this process extremely easy and is accomplished through an analysis engine.

We've implemented several common compression algorithms that run within HIP's framework to be executed on a SunSPOT processor. Figure 7 shows the results of a basic compression algorithm compared to raw data. This algorithm only sends data when a data point changes more than a predefined threshold from the previously sent data point. While the algorithm is simple, its results demonstrates the ease of analysis provided by HIP.

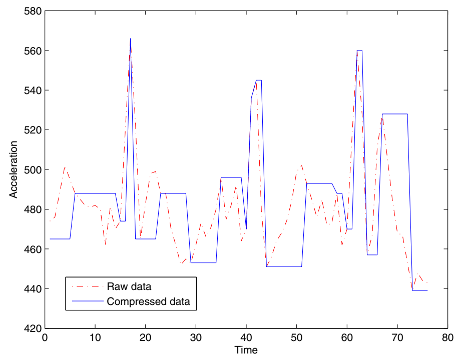

Figure 7. Live data is compressed using a simple compression algorithm. HIP allows for both compressed and raw data to be to be sent wirelessly to analyze the correctness and performance of a system in real time.

C. System Simulations

Testing complete end to end systems without a full deployment on patients is one challenge in wireless health applications. This issue is exacerbated by the need for medical trial approvals from one to many government organizations. We also must be sure to limit as much harm or discomfort as possible to patients (including humans and animals). However, with the proper framework, development and test can be done by using pre-existing data without the need for test subjects. In one such project, we were required to gather EMG data from a subject's forelimbs to detect walking patterns.

Using HIP, we were able to create a prototype trigger system without using a single test subject. Instead, we used pre-gathered data taken from previous experiments. We created a simulation application from our PC that wirelessly transmitted the raw data over a 2.4 GHZ band. We implemented a basic pattern recognition algorithm for walk detection on the receiving embedded platform. Once the device is ready for live testing, we can seamlessly switch off the wireless transmission and turn on live sensor data.

VI. Future Work

As we go forward with HIP, there are several features that we plan to implement. First, we would like to implement a series of libraries that plug into HIP's framework. One of these libraries will include a security suite. Security is extremely important when dealing with medical data [11][17]. We plan to release a suite of security protocols such as RSA and Elliptic Curve Cryptography. We also plan to release a complete compression library including common compression techniques including both lossy and lossless algorithms.

VII. Conclusion

This paper presented HIP - Health Integration Platform. Our work on HIP grew from necessity of a complete and generic wireless health platform. This paper presented the overall architecture and our experiences developing within HIP's framework. HIP has proven to be both effective and generic. We created HIP to encourage and facilitate further collaboration in the wireless health arena. HIP was specifically built with the ability to seamlessly integrate into pre-existing projects. HIP also supports a modular design to facilitate code sharing and accurate analysis.

References

[1] Au, L. K., Wu, W. H., Batalin, M. A., Mcintire, D. H., Kaiser, W. J.: MicroLEAP: Energy-aware Wireless Sensor Platform for Biomedical Sensing Applications. Biomedical Circuits and Systems Conference, BIOCAS 2007, November 2007, Montreal, Canada, 158-162 (2007)

[2] Bierl L.. MSP430 Family Mixed-Signal Microcontroller Application Reports. Texas Instruments, 2000.

[3] Delicato, F.C., Pires, P.F., Pinnez, L., Fernando, L., da Costa, L.F.R. "A flexible web service based architecture for wireless sensor networks," Distributed Computing Systems Workshops, 2003. Proceedings. 23rd International Conference on , vol., no., pp. 730-735, 19-22 May 2003.

[4] Heinzelman, W. R., Kulik, J., and Balakrishnan, H. 1999. Adaptive protocols for information dissemination in wireless sensor networks. In Proceedings of the 5th Annual ACM/IEEE international Conference on Mobile Computing and Networking (Seattle, Washington, United States, August 15 - 19, 1999). MobiCom '99. ACM, New York, NY, 174-185.

[5] Iyengar, S., Bonda, F. T. , Gravina, R., Guerrieri, A., Fortino, G., Sangiovanni-Vincentelli, A. 2008. A Framework for Creating Healthcare Monitoring Applications Using Wireless Body Sensor Networks. Proceedings of the ICST 3rd International Conference on Body Area Networks. Tempe, Arizona, 2008.

[6] Jovanov E., Price J., Raskovic D., Kavi K., Martin T., and Adhami R. 2008. Wireless Personal Area Networks in Telemedical Environment. In Proc. 3rd Int. Conf. Inf. Technol. Biomed. Arlington, VA, Nov. 2000, 22-27.

[7] Lo, B., Thiemjarus, S., King R., and Yang, G.Z. 2005. Body Sensor Network - A Wireless Sensor Platform for Pervasive Healthcare Monitoring. In Adjunct Proceedings of the 3rd International Conference on Pervasive Computing, May 2005, Munich, Germany, 77-80.

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

[9] Machine SUIF, Harvard EECS, http://www.eecs.harvard.edu/hube/software/

[10] Malan, D., Fulford-Jones, T., Welsh, M., Moulton, S.. 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.

[11] Miller, S. K. 2001. Facing the Challenge of Wireless Security. Computer 34, 7 (Jul. 2001), 16-18.

[12] Network Simulator 2 (ns2), http://www.isi.edu/nsnam/ns/

[13] Sadler, C. M. and Martonosi, M. 2006. Data compression algorithms for energy-constrained devices in delay tolerant networks. In Proceedings of the 4th international Conference on Embedded Networked Sensor Systems (Boulder, Colorado, USA, October 31 - November 03, 2006). SenSys '06. ACM, New York, NY, 265-278.

[14] Scalable Network Technologies Inc., "Qualnet Simulator", http://www.scalable-networks.com

[15] Simon, D., Cifuentes, C., Cleal, D., Daniels, J., White, D. Java on the Bare Metal of Wireless Sensor Devices: The Squawk Java Virtual Machine, Virtual Execution Environment (VEE), 2006.

[16] van Halteren, A.T. and Bults, R.G.A. and Wac, K.E. and Konstantas, D. and Widya, I.A. and Dokovski, N.T. and Koprinkov, G.T. and Jones, V.M. and Herzog, R. (2004) Mobile Patient Monitoring: The Mobihealth System. The Journal on Information Technology in Healthcare, 2 (5). pp. 365-373.

[17] Warren, S., Lebak J., Jianchu Yao, Creekmore, J., Milenkovic, A. Jovanov, E., Interoperability and Security in Wireless Body Area Network Infrastructures. Engineering in Medicine and Biology Society, 2005. IEEE-EMBS 2005. 27th Annual International Conference of the , vol., no., pp.3837-3840, 2005

[18] Woodbridge, J., Nahapetian, A., Noshadi, H., Sarrafzadeh, M., Kaiser, W. Wireless Health and the Smart Phone Conundrum. The 2nd Joint Workshop On High Confidence Medical Devices, Software, and Systems (HCMDSS) and Medical Device Plug-and-Play (MD PnP) Interoperability. April 2009, San Francisco, CA.