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

Application-Oriented Programming Model for Sensor Networks Embedded in the Human Body

Talles M. G. de A. Barbosa, Iwens G. Sene Jr, Adson F. da Rocha, Francisco A. de O. Nascimento, Hervaldo S. Carvalho and Juliana F. Camapum

***Abstract*—This work presents a new programming model for sensor networks embedded in the human body which is based on the concept of multi-programming application-oriented software. This model was conceived with a Top-Down approach of four layers and its main goal is to allow the healthcare professionals to program and to reconfigure the network locally or by the Internet. In order to evaluate this hypothesis, a benchmarking was executed in order to allow the assessment of the mean time spent in the programming of a multi-functional sensor node used for the measurement and transmission of the electrocardiogram.**

***Keywords*—deployment-time programmability, run-time reconfigurability, application-oriented software, transparency, wearable systems, BSN, BAN, Java.**

I. Introduction

The inherent characteristics of the Body Sensor Networks (BSN), like wireless communication and autonomous operation have a great potential to influence clinical applications for the next decades by promoting a new paradigm of healthcare based on wearable systems.

The goal of the Body-Worn Sensor Networks (BWSNET) is to build an infrastructure for monitoring the human health through wireless sensor networks embedded into the user's (patient) clothes and even in the body.

An application-oriented[1] software architecture, called SOAB (*Software Architecture for Body-Worn Sensor Networks Project*), has been developed in order to allow programming (at deployment-time) and reconfiguration (at run-time) support to a sensor network used to monitor the human body.

Manuscript received April 24, 2006. This work was supported by the National Council for Scientific and Technological Development of Brazil (CNPq), /2003–1, and by FINATEC (Fundacao de Empreendimentos Cientificos e Tecnologicos).

T. M. G. de A. Barbosa and I. G. Sene Jr. are with the Department of Computer Science of the Catholic University of Goiás, Goiânia, GO 74605-110 - Brazil (e-mail:talles@ucg.br and iwens@ucg.br).

A. F. da Rocha, F. A. de O. Nascimento and J. F. Camapum are with the Department of Electrical Engineering of the University of Brasília, 70919-970 Brasil (phone: +55 61 2735977; fax: +55 61 2746651; e-mail: assis@unb.br , adson@unb.br and juliana@ene.unb.br).

H. S. Carvalho is with the Medical School and with the Department of Electrical Engineering of the University of Brasília (e-mail: carvalho@unb.br).

[1] The term application-oriented is introduced in this paper to characterize systems projected or customized for a group of applications with similar and specific requirements, like a BSN. A WSN must be projected from requisitions of one or a group of similar applications, its Design Space [5]. Therefore, it should be application-oriented.

The importance of this work is related to the hypothesis that the self-adjustable behavior of a BSN together with the possibility of human intervention can result in the improvement of the metrics of the performance and the efficiency of these systems. As an example, the methodology may increase the life span of a network [1] [2] as well as its robustness (fault-tolerance).

Although there are several projects that employ RSSF to monitor the human health [3]-[20], most of them have not discussed the importance of specific models for BSN programming and reconfiguration until now. They have delegated this problem to a generic model based on the TinyOS *framework* [4] that does not offer proper support to the healthcare professionals, who are the traditional managers of these systems.

Furthermore, the projects that use the concept of multi-programming in the sensor-nodes do not work with the possibility of these nodes being multi-functional. A multi-functional sensor-node could allow different functionalities to run at different periods of time.

The SOAB was conceived based on a four layer *Top-Down* model (see Fig. 1b), that was designed so that healthcare professionals are able to program and reconfigure the network locally or using the Internet.

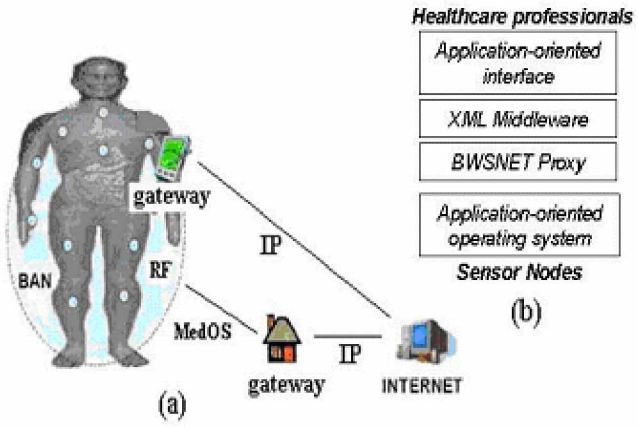

Fig. 1. (a) The use of a scenario for SOAB architecture. (b) The SOAB four layer model.

A case study involving programming and reconfiguration of a multi-functional sensor-node used to capture the ECG is presented in the next sections.

II. DESCRIBING THE SOAB

In the first layer of the SOAB, we have the *BWSNET Configuration Tool*. Due to its application-oriented nature and to the fact that it is not a general-purpose entity, this graphical interface is responsible for providing means by which the healthcare professionals, who will be the actual programmers from now on, can describe their algorithms without too much effort, meaning that it is less *time-consuming*, less *error-prone* and more intuitive. Moreover, it is a visual tool, which does not require specific knowledge of computational models.

This interface has also the ability to include new sensors that were not initially specified without the need to recompile the source code. To achieve that, an interface based on Java Reflection [6], with support to java applets, java applications and java thinlets [7] was provided. It should also incorporate the data model to a generic sensor-node as defined in [8].

The BWSNET Configuration Tool has also a simulator, so the programmer can evaluate the life span of the sensor-node, from the estimated amount of energy consumed from the nature of the tasks that were selected. Such a tool allows the programmer to evaluate the performance of the chosen configurations before the sensor-node programming is in fact initiated.

Fig. 2 illustrates the programming of the ECG sensor-node. Let us suppose that the programmer has selected a configuration for the placement of the electrodes. In this case, the programmer will have an ECG sensor-node with three different functionalities available: (i) ECG1 at 100 Hz, (ii) ECG3 at 500 Hz and (iii) ECG6 at 1000 Hz.

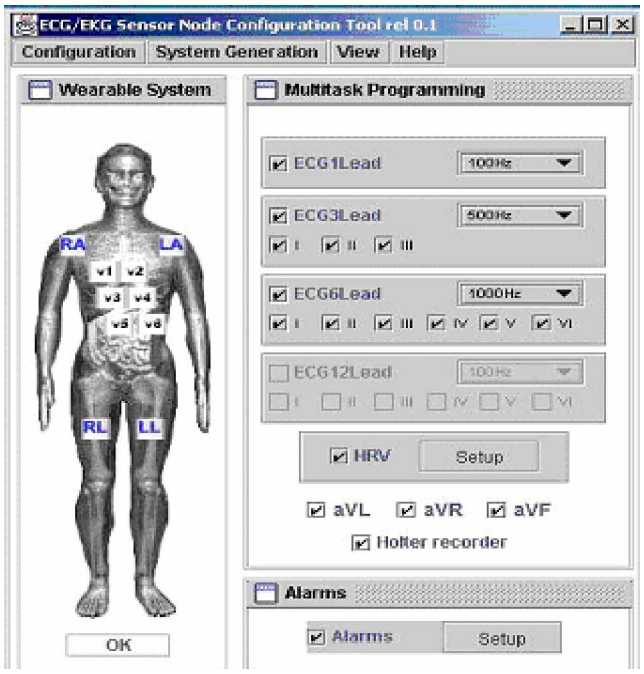

Fig. 2. The ECG graphical interface. Notice that the of the states of the automatum refers to the selection of tasks in the multitask programming palette.

The immediate benefit of this approach is related to the decrease in energy consumption derived from the changes of state of the automatum that describes the system behavior. The selection of the alarm can have influence in the programming of the events, even though the programming of the events is transparent to the programmer of the automata creator module. The state diagram that represents the behavior created with this configuration can be seen in Fig. 3.

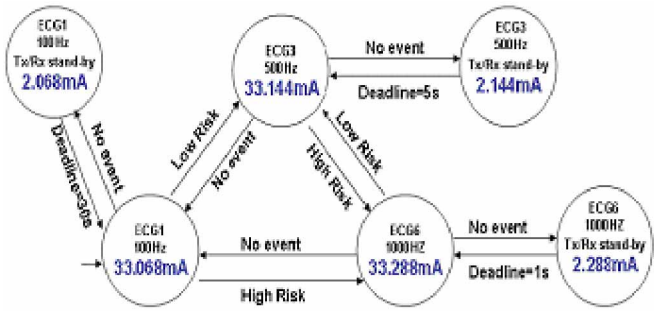

Fig. 3. The state diagram corresponding to the configuration presented in Fig. 2. This diagram represents the current consumption based on each feature that is executed at each instant of time. Besides the reduction in the operating frequency, a policy for shut down of the radio transmitter is associated to each feature. The values presented here refer to the hardware (sensor-node) described in section III. Further details about this modelling can be seen in [1].

Besides, if it becomes impossible for the system to operate with one of the three functionalities, there is still a chance of around 66% that the system remains active in one of the two remaining functionalities, considering the tasks to be completely independent.

In order to offer support to the programming and to the reconfiguration of the sensor-nodes from the Internet, a group of remote procedure calls (RPC) has been implemented in accordance with the recommendations of the W3C Web Services [9]. We believe that the middleware based in XML is the more flexible and generic way to interconnect a Body Area Network (BAN) with limited RF range to the programmers responsible for the management of the wearable sensor network, when there is an IP network between some of these points, see Fig. 1a.

In the third layer we have the *BWSNET Proxy*. This software entity was projected to operate in the gateway devices, see Fig. 1a. It is responsible for the translation of the requisitions of the format SOAP XML [10] to commands that are understandable to the underlying layer. Besides, it has the responsibility to create the real image of the system from an abstract model based on automata, the self-adjustable binary code that will be executed by the sensor-nodes. Another functionality from the Proxy server, equally important, is to promote the integration of the BSN with an external healthcare information system (HIS).

There is an application-oriented operating system running inside the sensor-nodes. It can be seen as the last layer of the SOAB architecture. MedOS is a command interpreter that

acts as a virtual machine upon the FreeRTOS operational system [11].

MedOS has as its main objective to promote the behavioral adjustment of the sensor-node at run-time, changing the priority values that can be associated with the functionalities provided by the sensor-node. The functionalities are represented by tasks created together with the libraries of the FreeRTOS.

The Java technology was used for the implementation of the three first layers of the SOAB, due to its portability and its adequacy to the methodology applied for the modeling and for the description of the software artifacts. To build the MedOS, we have used the C and C++ languages with the MSPGCC compiler [12].

III. Performance Evaluation and Results

In a first evaluation of the performance, a *benchmarking* was executed in order to assess the response time of the deployment-time programmability. The response time is a metric that can be help the assessment of user satisfaction with the service offered through the Internet. According to [13], for electronic commerce, values above 10 s can cause lack of interest for the service in the majority of the users.

The response time of this project corresponds to the deployment cycle time and it is composed by five main values: (1) elapsed time for the Proxy to receive the service requisition (SOAP message). The values of this metric are modified in accordance to the traffic conditions of the Internet. Also (2) the processing time of the service messages that corresponds to the *parser* of the SOAP messages. Its objective is to decode the automatum defined by the users, as the operations related to this automatum: (3) compile, (4) deploy and (5) run.

The compile operation corresponds to the automatic creation of the automatum that will represent the functionalities (tasks) described by the programmer, that means, the elapsed time in the creation of the binary file from the code libraries. The deployment operation corresponds to the elapsed time in the programming of the flash memory of the microcontroller and the operation run corresponds to the restart time.

The httperf tool [14] was used to simulate the operations executed from the *BWSNET Configuration Tool*. It was executed in a PC Pentium VI 2.4GHz 512 Mbytes RAM. The httperf is a program commonly used to evaluate the performance of Web servers. Besides the generation of the workload, the httPerf make the statistical reports available for analysis.

The *BWSNET Proxy* was executed in a PC Pentium III 700 MHz with 128 Mbytes RAM. Its purpose is to emulate the performance achieved when this artifact was being executed by a mobile device of less computational power. Regarding the link, an ETHERNET 10/100baseT network, without traffic isolation, in order to approximate as much as possible the real model, was used.

The sensor-node was assembled from the Olimex MSP430-HG439 kit [15], with a bluetooth radio BlueSMiRF Basic [16]. The circuits to capture the ECG were built based on the differential amplifier INA321 [17] and connected to the microcontroller like the one described in the project presented in [18]. Table 1 presents a summary of the main results achieved.

TABLE I
Response Time obtained for Programming the ECG sensor-node through The Internet

Code size (rounded)	Deployment cycle duration (rounded)	Binary code content description
5KB	4s	Just the FreeRTOS
6KB	5s	FreeRTOS + MedOS
9KB	10s	FreeRTOS + MedOS + ECG1/100Hz + ECG3/500Hz + ECG6/1000Hz

Deployment cycle duration corresponds to the rounded medium response time (in seconds) obtained from 10 interactions between the Httperf and the BWSNET Proxy. For each ECG task a 17-tap FIR low-pass filter with a cutoff frequency of 0.1 Hz was included, as well as a 60 Hz notch filter. The implementation occupied approximately 1 kB.

IV. Discussion

The values of the response time tend to increase according to the code complexity and to the inherent level of intelligence inside the sensor-node. They can also have influence in the amount of daily reprogramming that is necessary. In practice, the values presented in Table 1 should be considered as the period of time that the patient will have to wait with an inactive system and connected to the Proxy through the interface JTAG.

Considering another approach, where the programming at deployment-time would be done from a serial interface and from the *bootstrap loader* [19], the process becomes even slower and consumes more energy. Moreover, the amount of write operations in the Flash memory can determine the life span of the system [20].

The fact is that, not considering the possibility of reprogramming due to software or hardware faults, the health condition of the patient can determine the frequency of the system reprogramming. In many cases, this is achieved by doing a slightly behavioral readjustment in the software at run-time. To do that, a complementary approach has been developed to allow the reprogramming to be executed without the need for interruption of the system neither the connection of cables.

The run-time reconfiguration methodology will allow the programmers to act not only in the sensor network level but also in the sensor-node level.

At the network level, it will be possible to define the importance of each sensor-node in accordance to the need of each application, defined by [2] as Sensor QoS Level (see Fig. 4). Sensor QoS Level refers to the importance of the information flux collected by a certain sensor-node. In this case, the value 0,8 says that the information flux expected

for this sensor-node is approximately 80% of the maximum flux. The adjustment of the size of the flux is directly related to the adjustment of the bandwidth and of the transmission rate used by the sensor-node at each instant of time.

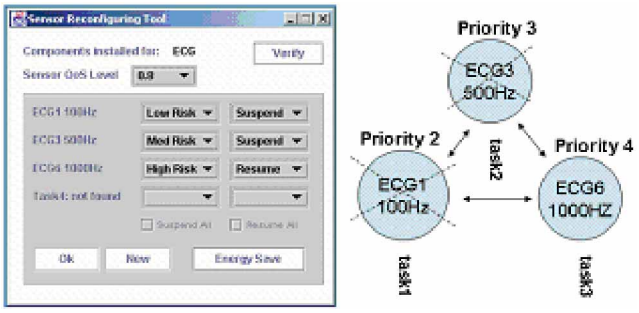

Fig. 4. The remote reconfiguration process and its implications at sensor level.

At the sensor-node level, a priority value will be associated to each executed functionality. The changing in the values of the priorities can modify the sequence, when no event occurs and, mainly, the time-slice that each task will use the CPU. In Fig 4. three priority values are used: Low Risk, Medium Risk e High Risk which are mapped respectively in three priority values by the MedOS: Priority 2, Priority 3 and Priority 4.

The programmer will be able to suspend functionalities that are less interesting in favor of others of greater interest. Furthermore, he can modify the priority values without any change in the autonomous operation capacity of the sensor-node, because the events defined at deployment-time will be kept.

V. Conclusions and Future Works

The concept of application-oriented software presented in this paper refer to the ability of the interfaces to adapt themselves to the profile of the possible users.

This paper presents a model for the programming of BSNs taking into account the fact that the programmers of these systems need tools with more transparency to carry out their activities such that these activities are not inconvenient to their patients.

The transparency here presented is evaluated by means of a benchmarking with the objective of quantifying how inconvenient this process can become even when BSNs projected to operate autonomously are used.

A software architecture in layers allows the transparency to be achieved in a modular way, thus protecting the portability related to the possible modifications of the underlying layers, including the hardware itself.

Future efforts will be concentrated on: (i) the implementation of mechanisms for security improvement and authentication, mainly for the remote interventions over the Internet; (ii) new algorithms based on priorities and application-oriented algorithms that can be used to get more efficient scaling of the functionalities embedded in the sensor-nodes, with the aim to guarantee the absence of deadlocks and starvation.

References

[1] H. S. Carvalho et al., “Efficient power management in real-time embedded systems”, presented at the 9th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA'03), Lisbon, Portugal, 2003.

[2] H.S. Carvalho, M. Perillo, W. Heinzelman and A. Murphy, "Middleware to support sensor network applications," *IEEE Network Magazine,* Special Issue, Jan. 2004.

[3] G. Yang, “Body Sensor Networks” [Online]. Available: http://ubimon.doc.ic.ac.uk/bsn/index.php?m=206

[4] UC Berkeley, “TinyOS” [Online]. Available: http://www.tinyos.net/

[5] K. Römer, R. Mattern, “The design space of wireless sensor networks”, *IEEE Wireless Communications*, vol. 11, no. 6, pp.54-61, December 2004.

[6] D. Green, “Trail: The Reflection API” [Online]. Available: http://java.sun.com/docs/books/tutorial/reflect/

[7] R. Bajazat, “Thinlet” [Online]. Available: http://thinlet.sourceforge.net/home.html.

[8] H. S. Carvalho, W. Heinzelman, A. Murphy and C. Coelho, "A General Data Fusion Architecture", presented at International Conference on Information Fusion (Fusion 2003), July 2003 [Online]. Available: http://www.ece.rochester.edu/~wheinzel/

[9] W3C. “Web Services Architecture” [Online]. Available: http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/

[10] W3C. “SOAP Version 1.2 Part 1: Messaging Framework” [Online]. Available:http://www.w3.org/TR/2003/REC-soap12-part1-20030624/

[11] R. Barry, “FreeRTOS” [Online]. Available: http://www.freertos.org/

[12] D. Dirky and C. Liechti, “The GCC toolchain for the Texas Instruments MSP430 MCUs” [Online]. Available: http://mspgcc.sourceforge.net/

[13] A. Bouch, “A user’s perspective of network QoS and changing”, PhD thesis, Depto. Of Computer Science, University College London, London, UK, 2001.

[14] D. Mosberger and T. Jin. “Httperf: a tool for measuring web server performance. *Performance Evaluation Review*, Volume 26, Number 3, December 1998, 31-37 [Online]. Available: http://www.hpl.hp.com/research/linux/httperf/docs.php

[15] Olimex. “MSP430-HG439 MPS430FG439 HEADER BOARD” [Online]. Available: http://www.olimex.com/dev/index.html

[16] Spark Fun. “Bluetooth Modem - BlueSMiRF Basic” [Online] .Available:http://www.sparkfun.com/commerce/product_info.php?products_id=158

[17] Texas Instruments. “INAA321, Micropower single-supply CMOS Instrumentation Amplifier” [Online]. Available: http://focus.ti.com/docs/prod/folders/print/ina321.html

[18] M. Raju, “Heart Rate and EKG Monitor using the MSP430FG439”, [Online].Available:http://focus.ti.com/docs/apps/catalog/resources/appnoteabstract.jhtml?abstractName=slaa280

[19] Texas Instruments. “Features of the MSP430 Bootstrap Loader (Rev. C)”[Online].Avaliable:http://focus.ti.com/docs/apps/catalog/resources/appnoteabstract.jhtml?abstractName=slaa089c

[20] H. Dai, M. Neufeld, R. Han, "ELF: An Efficient Log-Structured Flash File System for Wireless Micro Sensor Nodes", presented at 2nd ACM Conference on Embedded Networked Sensor Systems (SenSys), 2004,pp.176-187.Avaliable:http://mantis.cs.colorado.edu/tikiwiki/tiki-index.php?page=Publications