ANNUAL JOURNAL OF ELECTRONICS, 2009, ISSN 1313-1842

Profile-based protocol for data interactions in Body Sensor Networks

Mitko Petrov Shopov, Nikolay Rumenov Kakanakov and Galidiya Ivanova Petrova

Abstract - **The paper presents an application level protocol for data interactions in local networks of embedded devices and its adaptation for Personal Health Systems and Body Sensor networks. The protocol uses profiles to improve scalability and applicability and to ease the customization for different application areas. Profiles are described in XML to be easily processed and written. The profile can describe the device parameters and application-specific context. Finally, to demonstrate the use of profiles as an extension for describing the particular application of the protocol the scenario of body sensor network will be used.**

Keywords – **Body Sensor Networks, CNDEP, Profiles**

I. Introduction

The progress of the information technology and electronic equipment has become a base for development of Internet and electronic services in every part of the society. Adoption of e-health, e-learning, e-government, and other e-services is no longer ideology but has its realizations, based on Internet and Web technologies. These technologies are applicable for information and services interaction and provide pervasive user interface. The real application of these e-services depends not only on the user access and information flows on the upper layers. Technologies should be developed on the lower device levels where the real instrumentation works. It will allow bringing more e-services to the home of the user. In the case of e-health services users must not only access the information but to provide raw data to the medical personnel. In many currant e-health services user can provide its physiological and contextual parameters (e.g. ECG, EEG, Heart and respiration rates, blood pressure, oxygen saturation (SpO2), body temperature, glucose level, spatial location and etc.) filling forms. The problem is how user will measure these parameters without instrumentation and medical control. The use of Personal Health Systems (PHS) and Body Sensor Networks (BSN) provides a solution to this problem. The interaction of PHS with medical servers and users will follow the ideology of e-services and Web technologies, but appropriate protocols for data interaction in BSN should be developed and appropriate communication technologies should be chosen. There are some realizations of the lower level data interaction based on industrial protocols or adopted form Web. A specific profile for medical devices is added in CANOpen [1] but the CAN communication is not applicable in BSN because it has no wireless realization. A Device Profile for Web Services [2] provides a communication independent framework for data interaction, but the use of SOAP requires more memory and processing power.

In this paper an application level protocol for data interaction in Personal Area Networks and BSNs is presented. The protocol is developed for data communications in controller networks and is based on profiles to be easily adaptable in different application areas. Profiles define application scenarios and device parameters. As an example a profile for personal healthcare is presented.

II. Profile-based version of CNDEP

A. CNDEP – general information and communication scenarios

CNDEP is a protocol for control and monitoring in TCP/IP based networks of embedded devices. It resides on the application layer of the TCP/IP protocol stack and uses UDP as a transport protocol. CNDEP is an asymmetric, byte-oriented protocol that exchange messages between embedded devices. Messages contain sequential bytes representing data and metadata, encoded in ACSII format. Message exchange is based on request / reply mode except for publish/subscribe mode where on one request multiple replies are given back. The main functionality is resided in the server side but error processing, packet loss and complex data processing are resided in the client. The idea is that the server-side is embedded device or smart sensor/actuator with limited processing power [4].

The protocol is designed to reflect the most typical deployment scenarios applicable for controller networks (figure 1). The four basic scenarios are [3]:

a) polling: master can request particular measurement (in case of sensor) or control (in case of actuator);
b) configuration: master can set configuration parameters concerning slave device operation;
c) alarm handling: slave device could be configured to generate alarm messages in case of error or occurrence of particular condition;
d) continuous monitoring: the master requests measurement information once and starts to collect continuous time series;

Implementation of these scenarios requires three kinds of communication services:

M. Shopov is with the Department of Electronics, Faculty of Electronics and Automatics, Technical University – Sofia, Plovdiv branch, 61 St. Petersburg Blvd., 4000 Plovdiv, Bulgaria, e-mail: mshopov@tu-plovdiv.bg

N. Kakanakov is with the Department of Computer Systems and Technologies, Faculty of Electronics and Automatics, Technical University – Sofia, branch Plovdiv, 61 Sankt Petersburg Blvd., 4000 Plovdiv, e-mail: kakanak@tu-plovdiv.bg

G. Petrova is with the Department of Electronics, Faculty of Electronics and Automatics, Technical University – Sofia, Plovdiv branch, 61 St. Petersburg Blvd., 4000 Plovdiv, Bulgaria, e-mail: gip@tu-plovdiv.bg

- request/response: this kind of communication can be used for scenarios a) and b)
- spontaneous: similar to request/response, but with changed roles between master and slave;
- subscription: an alternative to polling mechanism; after subscription request the roles of master and slave changes;

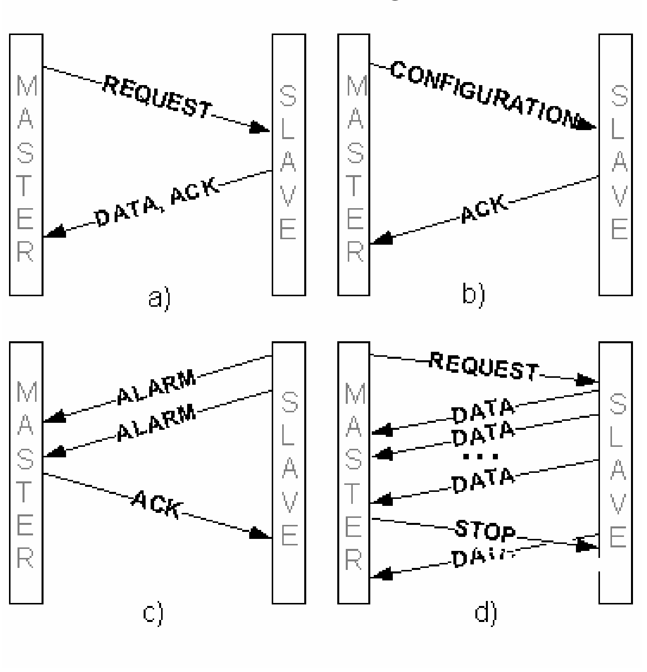

FIG. 1: SCENARIOS FOR REMOTE SERVICES: A) POLLING B) CONFIGURATION C) ALARM HANDLING D) CONTINUOUS MONITORING

The state transition diagram (STD) of the server-side protocol implementation is shown on fig. 2 and the state transition conditions (triggers) are shown on table 1. After power on or hardware reset the protocol enter in initialization state. In this states prerequisites for the protocol work are carried out as network and socket preparation, device advertisement and etc. After successful initialization the protocol enters the IDLE state in which it stays most of the time. In this state the server side listens for the incoming messages or waits for events connected to device's work or changes in the environment. The IDLE state can be designed for energy efficiency if needed. Incoming messages are decoded (state DECODE) to define its category (simple or containing data) and command type (test, read, write, execute, register/unregister subscriber, clear alarm).

These different command types define different states. Test command is used (if appropriate) for keep-alive messages that master uses to check devices' presence in the network. If a specific event appears in the device it broadcasts alarm messages until it receives clear alarm command. Alarm events can be defined using relevant write command – they can be application or device specific. Read command is used to read internal data that can be set by write command (some specific values can be read-only e.g. device ID).

Execute command is used for specific control functions – it causes execution of command on the device. This execution can be restarting specific task on device, contacting sensor or actuator, reconfigure protocol parameters. Register/unregister subscriber is used for monitoring – client registers its address and the data it is interested in and when data changes server spontaneously sends data to client(s).

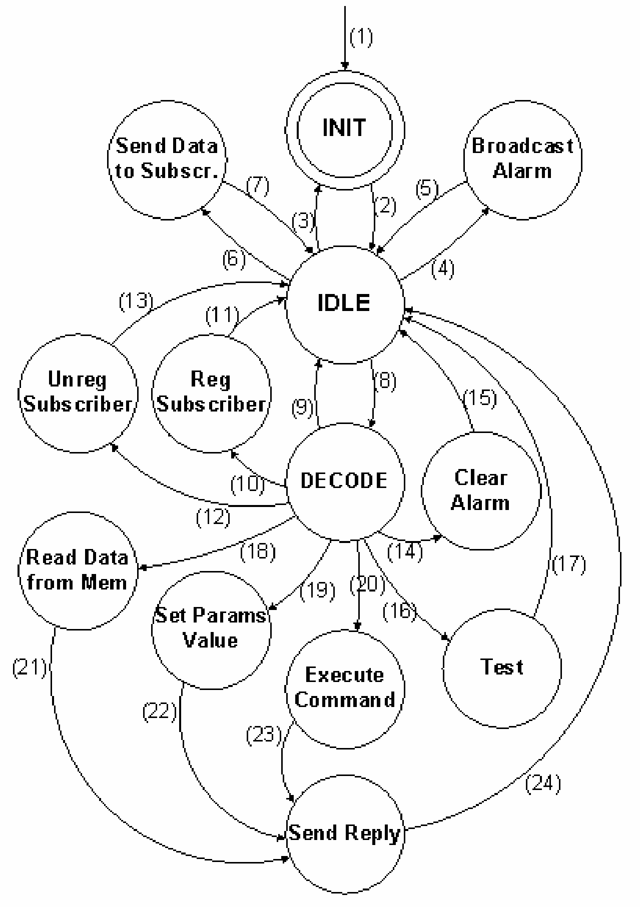

FIG. 2: SERVER-SIDE STATE-TRANSITIONAL DIAGRAM

When client no longer needs this data it uses unregister. There are two ways to read data from device – read or execute. The first way takes less time to proceed but in the latter more actual data is read on demand.

TABLE 1. TRIGGERS FOR STATE TRANSITION

(1), (3)	At Power on, HW reset, or SW restart the initialization state is entered autonomously
(2)	Initialization finished
(4)	Alarm condition detected
(6)	Subscription data ready
(8)	Request received
(10)	Subscribe request received
(12)	Unsubscribe request received
(14)	Alarm condition acknowledged
(18)	Request for data from memory
(19)	Request to write data to memory
(20)	Request for execution of command on the embedded device
(21),(22),(23)	Request processed – send appropriate reply message
(5),(7),(9),(11), (13),(15),(17), (24)	State's branch processing finished, return to listening for requests and interrupts from timers and alarms

B. CNDEP Profiles

The CNDEP protocol uses *profiles* to improve its scalability and applicability and to ease the customization for different application areas. Profiles are described in XML to be easily processed and written. The availability of XML scheme gives provisions for high-level validation in the design state of a protocol profile. The profile can describe the device parameters and application-specific context.

Several profiles can be merged to form one complete profile. To avoid duplications, deletions and other similar errors profiles are separated vertically by a parameter called level. The rule is that profiles with higher level numbers have precedence and their rules and commands cannot be replaced by profiles with lower level numbers.

The structural description of CNDEP protocol in XML profiles along with the possibility for strict validation using XML schemes gives possibility for development of a development tool for automatic code generation producing the skeleton of the application with blank application function to be filled by embedded programmers.

Distributed systems for control and monitoring typically have hierarchical structure (figure 3). In such networks communications can be described in two groups – one for system identification and configuration, and one for monitoring and control of the environment.

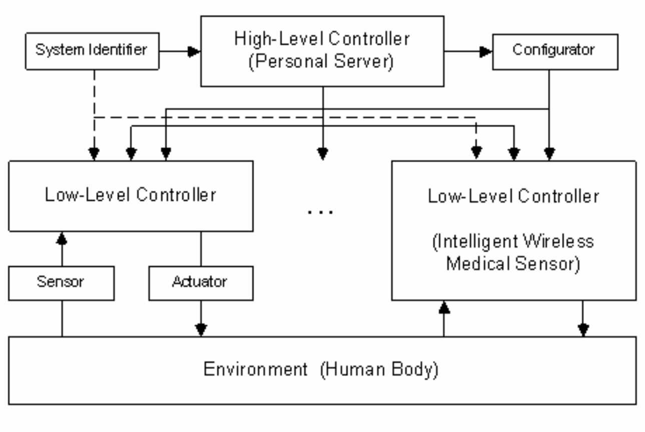
System Identifier
High-Level Controller (Personal Server)
Configurator
Low-Level Controller
...
Low-Level Controller (Intelligent Wireless Medical Sensor)
Sensor
Actuator
Environment (Human Body)

FIG. 3: EXAMPLE HIERARCHICAL STRUCTURE OF A PERSONAL HEALTHCARE DISTRIBUTED SYSTEM

In the CNDEP profile these groups are reflected by *deviceProperties* and *applicationProperties*. The following fragment of XML general profile illustrates this:

```
<profile level="0" name="general">
  <group name="deviceProperties">
   ……
  </group>
  <group name="applicationProperties">
   ……
  </group>
</profile>
```

Each of these groups can contain multiple commands. Some commands are more complex than others and can contain sub-commands. Commands and sub-commands consist of description, one or more requests and responses. Commands and sub-commands can be of any of the following types:

```
  <xs:enumeration value="read" />
  <xs:enumeration value="write" />
  <xs:enumeration value="execute" />
  <xs:enumeration value="publish" />
  <xs:enumeration value="subscribe" />
```

The description field should be useful in automatic code-generation development tools. Request and responses has *category* and *context type* attributes used for their interpretation at the opposite side. Among with standard context types some application specific context type may be included (e.g. *degC* used to describe a floating point number representing degree Celsius).

III. Use-case Scenario

To demonstrate the use of profiles as an extension for describing the particular application of CNDEP protocol the scenario of body sensor network will be used. The scenario consists of ECG sensor equipped with Bluetooth interface [5], embedded platform based on Cirrus EP9302 microcontroller, industrial Bluetooth module [6], and Linux OS [7].

A. Body Sensor Network and Personal Server

Body sensor network (BSN) is a special case of wireless sensor networks. It consists of a set of miniaturized, low cost, wearable or implantable bio-sensors and actuators that are expected to provide continuous monitoring of the patient's physiological and contextual parameters (e.g. ECG, Heart rate, blood pressure, oxygen saturation (SpO_2), body temperature, glucose level and etc.). The network is usually coordinated by Personal server [8], [9].

B. Personal Server

Personal Server is responsible for transparent interface to BSN, interface to the user, and an interface to the medical server, performing high-level data processing, analysis and temporary local storage. The interface to BSN should deal with network configuration and management. The configuration tasks include sensor node registration, initialization, customization, calibration, and setup of secure communication. The management tasks include channel sharing, time synchronization, data retrieval and processing, and data fusion. The most adoptable platforms for realization of personal server are PDAs and smart cell phones due to their size, processing and communication capabilities. Other possibilities are tablet PCs, laptops, and custom specially designed microprocessor-based devices. [10], [11].

C. ECG CNDEP Profile

There is a profile called General that includes all the basic functions that are common to most embedded applications. This profile is of level zero and will not overwrite command definitions in higher level profiles. The ECG profile rely on the common command of General profile and include only commands specific to the particular Bluetooth-enabled ECG module – [5]. Some of the commands definitions in the profile however are not yet functionally supported by the module.

The following fragment illustrates definition of a command from deviceProperties group used to check the remaining battery charge of the ECG module.

```
<command name="Check battery status"
                  type="read" id="127">
  <request category="simple" />
  <response category="data"
                  contextType="percentage"/>
  <response category="error"
                  contextType="decimal" />
</command>
```

The applicationProperties group of commands includes the following commands:

- *Get ECG Data*: used to obtain a single sample of ECG data by the mechanism of polling;
- *Get Bulk ECG Data*: used to obtain several samples of ECG data at once. Useful for power-saving;
- *Get/Set Mode*: used to set the number of channels. If the user is interested in less than the maximum number of channels, the rest could be skipped. Also useful for power-saving;
- *Get/Set Precision*: set ADC precision (if possible). Less precision (if applicable) means less data, less transmissions, and less power consumption;
- *Subscribe/Unsubscribe*: users can issue this command to subscribe/unsubscribe for periodic ECG data;
- *Publish*: used by the ECG module to send subscription data to all registered clients.

Some of the commands' definitions are illustrated by the following fragment.

```
<command name="Get Bulk Data"
                   type="read" id="?">
  <description>Get several samples at once
      (power-save modes).</description>
  <request category="simple" />
  <response category="data"
                   contextType="multiECG" />
  <response category="error"
                   contextType="decimal" />
</command>

<command name="Set precision"
                    type="execute" id="?">
  <description>Set ADC Precision
             [number of bits]</description>
  <request category="data"
                    contextType="decimal" />
  <response category="ack" />
  <response category="error"
                    contextType="decimal" />
</command>

<command name="Subscribe" type="subscribe"
                                 id="?">
  <request category="data"
                 contextType="ipAddress" />
  <response category="ack" />
  <response category="error"
                 contextType="decimal" />
</command>
```

IV. Conclusions and Future work

In this paper an adaptation of a protocol for data interactions in local networks of embedded devices is presented. The introduction of profiles in protocol design should give more flexibility and scalability by allowing embedded applications' designers to easily customize and optimize the protocol to their needs. The XML description of the profiles makes it easily processed and written, and along with XML validation makes the protocol definition more error-proof and suitable for automatic generation of the code of its implementation files.

Some future work includes design and development of tools for automatic generation of code from XML profiles for some pilot embedded platforms and operating systems (if OS is available). Another area for future work includes extending the protocol design with new profiles for new type of sensors and actuators and application areas.

V. Acknowledgements

The presented work is supported by Technical University of Sofia, project "091ni060-17/2009" – entitled "Development of experimental patient monitor for data collections in body sensor networks" under contract "091ni060-17/2009".

References

[1] CiA 412-1,DS v1.0: *CANOpen Profiles for medical devices*, 2006.

[2] Jammes, F., H. Smit, *Service-oriented architectures for devices - the SIRENA view*, Proc. 3rd IEEE International Conf. on Industrial Informatics, pp. 140- 147, 10-12 Aug. 2005.

[3] Topp, U., P. Mueller. *Web based service for embedded devices*, Lecture Notes in Computer Science, Vol.2593, pp. 141-153, 2003.

[4] N. Kakanakov, I. Stankov, M. Shopov, and G. Spasov, *Controller Network Data Extracting Protocol – design and implementation*, Proc. CompSysTech Conference, V. Tarnovo, Bulgaria, 2006, pp.IIIA-14-1 – IIIA-14-6.

[5] Iliev I., Tsvetanov D., Matveev M., Naydenov S., Krasteva V., Mudrov N., *Implementation of high resolution wireless ECG data acquisition system in intensive coronary care unit*, Proc. International Conference Advanced Information and Telemedicine Technologies for Health '2005, Minsk, Belarus, pp. 79-84, November 2005.

[6] Parani SD 100 Bluetooth Serial Adapter Homepage – http://www.sena.com/products/industrial_bluetooth/sd.php.

[7] Port of Linux Debian OS for ARM – http://www.debian.org/ports/arm/.

[8] Park, S. and Jayaraman, S. , *Enhancing the Quality of Life Through Wearable Technology*, IEEE Engineering in Medicine and Biology, vol. 22, no. 3, pp. 41-48, May/June 2003.

[9] Lo, B. and Yang, G., *Body Sensor Networks – Research Challenges and Opporunities*, Proc. Antennas and Propagation for Body-Centric Wireless Communications, pp. 26-32, April 2007.

[10] Istepanian R., Jovanov, E., and Zhang Y.T., *Beyond Seamless Mobility and Global Wireless Health-Care Connectivity*, IEEE Transactions on Information Technology in Biomedicine, vol. 8, no. 4, pp. 405-441, Dec. 2004.

[11] Yu, P., Ming, X., Hui Yu, and Guo Q. Xiao, *The Challenges for the Adoption of M-Health*, Proc. IEEE Conf. on Service Operations and Logistics, and Informatics (SOLI '06), pp.181-186, June 2006.