An Appstore Framework for Body Sensor Networks

Andrea Gaglione
a.gaglione@imperial.ac.uk

Benny Lo
benny.lo@imperial.ac.uk

Guang-Zhong Yang
g.z.yang@imperial.ac.uk

The Hamlyn Centre, Institute of Global Health and Innovation
Imperial College London
London, SW7 2AZ, UK

***Abstract*—One of the main challenges in body sensor networks (BSNs) is the remote maintenance of practical deployments in human body environments. Next generation BSN applications ('micro-apps') will require new platforms that enable vendors and developers to distribute their applications through an online storefront and allow users to download them on their BSN deployments. In this paper, we introduce an appstore framework for BSNs that represents a first attempt to provide a comprehensive solution to this need. Our idea is similar to that of existing appstores for smartphones. The main difference in our case is that applications need to be downloaded on remote BSN deployments consisting of multiple devices, entailing the need for a reprogramming service to spread application images over the sensor nodes. We present our architecture and a working prototype that allows for uploading micro-apps to the appstore and downloading them on a single sensor node.**

I. Overview

Body sensor networks (BSNs) are expected to play a crucial role in the next generation pervasive healthcare, wellbeing and sport domains [1]. BSN applications, termed as **"micro-apps"**, are usually designed to serve a single purpose and operate for extended periods of time. However, the application requirements may vary, creating the need for reprogramming the nodes with new software. For example, in a context aware sensing scenario, we may need to change the way a patient is monitored depending on specific conditions (e.g. day/night, patient sleeping in the room or moving around). Unlike the traditional method of programming the nodes over a dedicated serial link, the embedded nature of these systems, often consisting of permanent implants in the human body, requires the propagation of new code over the network. The real strength of using an "over-the-air reprogramming" technique [2] is that micro-apps can be maintained remotely and devices can evolve autonomously, without the need for nurses or clinicians to reconfigure and adapt them to fulfill the new functionality.

We envision a scenario where *application developers* distribute their applications through an online storefront, which then allows *domain experts* (end users, e.g. medical doctors, technicians, nurses) to check the list of available applications and download them on their BSN deployments. Figure 1 illustrates a high level system architecture of the BSN appstore framework. The core component is the *appstore*, an online available application running on a server machine (*application server*), which keeps the information about the available micro-apps in a repository. End users may access the appstore through an Internet connection and download micro-apps on remote *BSN deployments*. In that case, the application image is sent from the application server to a *BSN gateway*, and then disseminated to the sensor nodes through a *reprogramming service*.

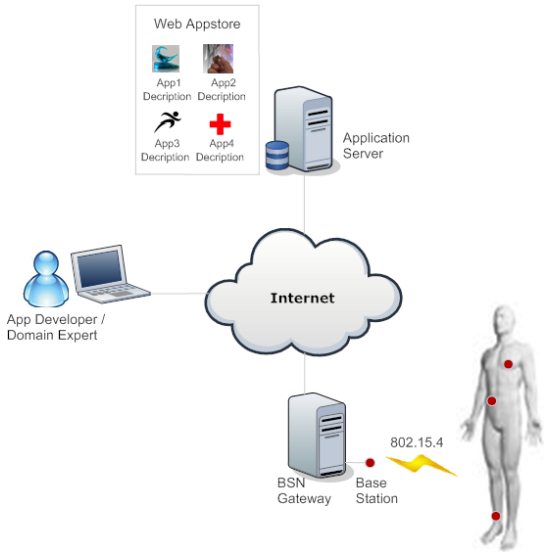

Fig. 1. High-level system architecture of the BSN appstore framework.

II. Design

The architectural model of the framework derives from [3–5] the capability to support the integrated management of multiple deployments (e.g. several patients monitored by medical staff). Figure 2 shows the architectural details of the main components of the framework. The appstore application consists of four main modules: a *user interface*, a *deployment manager*, an *apps manager*, and a *dispatcher*. Both application developers and domain experts access the system over the Internet and interact with it through its user interface. The deployment manager keeps information about BSN deployments (e.g. BSN gateway address, node hardware) in a *deployment DB*. Any request to update the deployment DB is processed by the *configuration handler*. The apps manager maintains a list of micro-app binary images, by keeping their information in the *apps DB* and mainly allows to: (i) upload images to the

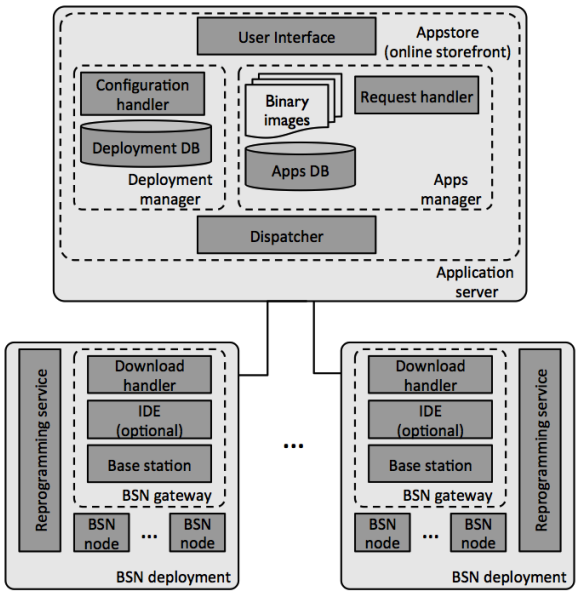

Fig. 2. Architectural details.

system and (ii) download images on the BSN deployments. The *request handler* is in charge of executing such tasks. In case of download, it triggers the dispatcher to send a micro-app binary image to a specific BSN deployment.

The access point of each BSN deployment is the related BSN gateway. It is typically a PC-class device connected to a *base station*, which collects sensor data from the *BSN nodes*. The *download handler* is an active process on that machine, which receives the binary images, stores them locally, and triggers the download process on the underlying network. Optionally, the gateway can host an *IDE* (Integrated Development Environment), used to develop micro-apps and download them on the nodes. The IDE should provide an API used by the download handler to start image dissemination through a reprogramming service. If the IDE is missing, the download handler should directly access to the reprogramming service.

III. Current Status and Preliminary Evaluation

We have developed a working prototype to show the basic functionality of the framework. We have implemented the main modules that allow for uploading micro-apps to the appstore and downloading them on a single sensor node, connected to the application server machine. We keep the list of binary images of the available micro-apps in a folder of the application server. The implementation of the deployment manager and the apps DB is underway as well as the integration of a reprogramming service into the framework. A single sensor node is directly connected to the application server, while the request handler is a server process just sitting on a port and waiting for client connections. We have developed a client application with a command line interface that allows users to send upload and download requests to the server. We target the BSN node [6], a platform specifically designed for BSNs, running micro-apps compiled for the BSNOS [7, 8].

The download handler is a process running on the server machine that triggers the download process through the BSNOS IDE. The latter aims to simplify the coding of applications and their download on the nodes, and is available online [9] as part of the BSNOS platform. In the current prototype, the download handler is directly invoked by the request handler. All the components are written in the Java programming language, just like the BSNOS IDE, in order to easily exploit its API. We used Java networking and threading libraries to implement a socket-based TCP/IP protocol between the application server process and the client application.

We set up a small testbed with a single BSN node, a 32-bit PC class device as the application server, and a user laptop. The BSN node is connected to a serial port of the application server, which also acts as the gateway of our single BSN node. The client application runs on a laptop and is able to open connections to the server on a specified port where the request handler is sitting on. We have developed a few simple micro-apps through the BSNOS IDE and compiled them to BSNOS bytecode, targeting the BSN node platform [6]. The latter is a hardware platform, equipped with the TI MSP430F1611 MCU [10] and specifically designed to ease the development of pervasive healthcare systems. We successfully tested the main functionality of the prototype and verified the execution of an upload task and a download task as well as a concurrent execution of both. Source files and binary images of the micro-apps used in the tests are available online [11].

We are currently extending the prototype and developing the whole appstore functionality. We are integrating a reprogramming service into the framework and planning a comprehensive evaluation of its actual benefits. The framework could potentially support medical and specialized staff in sharing and deploying micro-apps, as well as in maintaining their BSN deployments.

References

[1] B. Lo, S. Thiemjarus, R. King, and G.-Z. Yang, *Body Sensor Network - A Wireless Sensor Platform for Pervasive Healthcare Monitoring*. In PERVASIVE (2005).

[2] J. W. Hui and D. Culler, *The Dynamic Behavior of a Data Dissemination Protocol for Network Programming at Scale*. In SenSys (2004).

[3] V. Casola, A. Gaglione and A. Mazzeo, *A Reference Architecture for Sensor Networks Integration and Management*. In GSN (2009). Springer LNCS.

[4] F. Amato, V. Casola, A. Gaglione, and A. Mazzeo, *A Common Data Model for Sensor Network Integration*. In CISIS (2010). IEEE Computer Society.

[5] F. Flammini, A. Gaglione, N. Mazzocca, V. Moscato, and C. Pragliola, *On-line integration and reasoning of multi-sensor data to enhance infrastructure surveillance*. JIAS (2009).

[6] BSN node. Web page—http://ubimon.doc.ic.ac.uk/bsn/index.php?article=167.

[7] J. Ellul, B. Lo, and G.-Z. Yang, *The BSNOS Platform: A Body Sensor Networks Targeted Operating System and Toolset*. In SENSORCOMM (2011).

[8] BSNOS. Web page—http://sourceforge.net/projects/bsnos/.

[9] BSNOS IDE. Web page—http://sourceforge.net/projects/bsnoside/.

[10] MSP430F1611 platform. Datasheet—http://www.ti.com/lit/ds/symlink/msp430f1611.pdf.

[11] Example micro-apps. Web page—http://www.doc.ic.ac.uk/~agaglion/micro-apps.htm.