Managing Wearable Sensor Data through Cloud Computing

Charalampos Doukas
Dept. of Information & Communication Systems Engineering
University of the Aegean
Samos, Greece
doukas@aegean.gr

Ilias Maglogiannis
Dept. of Computer Science & Biomedical Informatics
University of Central Greece
Lamia, Greece
imaglo@ucg.gr

***Abstract*—Mobile pervasive healthcare technologies can support a wide range of applications and services including patient monitoring and emergency response. At the same time they introduce several challenges, like data storage and management, interoperability and availability of heterogeneous resources, unified and ubiquitous access issues. One potential solution for addressing all aforementioned issues is the introduction of the Cloud Computing concept. Within this context, in this work we have developed and present a wearable – textile platform based on open hardware and software that collects motion and heartbeat data and stores them wirelessly on an open Cloud infrastructure for monitoring and further processing. The proposed system may be used to promote the independent living of patient and elderly requiring constant surveillance.**

Keywords-component; cloud computing, sensors, wearable sensors, wearable data management

I. INTRODUCTION

The introduction of the pervasive healthcare paradigm has enabled the awareness towards the elderly and the need for constant medical supervision of chronic patients or habitants at remote, isolated and underserved locations. In this context, advanced electronic healthcare services are required to be made available through a network anytime, anyplace and to anyone. A medical assistive environment on the other hand concerns the utilization of pervasive and ubiquitous technologies for delivering the above services. Wireless technologies enable the real time transmission of data about a patient's condition to caregivers. Numerous portable devices are available that can detect certain medical conditions—pulse rate, blood pressure, breath alcohol level, and so on—from a user's touch.

The realization, however, of health information management through mobile devices introduces several challenges, like data storage and management (e.g., physical storage issues, availability and maintenance), interoperability and availability of heterogeneous resources, security and privacy (e.g., permission control, data anonymity, etc.), unified and ubiquitous access. One potential solution for addressing all aforementioned issues is the introduction of Cloud Computing concept in electronic healthcare systems. Cloud Computing provides the facility to access shared resources and common infrastructure in a ubiquitous and pervasive manner, offering services on-demand, over the network, to perform operations that meet changing needs in electronic healthcare application. **Figure 1** illustrates the Cloud Computing concept.

Figure 1. An illustration of the Cloud Computing concept. All kinds of computing and communication devices are able to interact with the Cloud and share the same data resources. Embedded - sensor devices and microcontrollers are such way a part of the Cloud.

In this context we have developed a Cloud-based system that manages sensor data. Wearable – textile sensors collect biosignals from the user (like heart rate and temperature) and motion data. Depending on the wireless technology used, the data can be forwarded to a mobile phone or directly to the Cloud

978-0-7695-4622-3/11 $26.00 © 2011 IEEE
DOI 10.1109/CloudCom.2011.65

infrastructure. The latter is built utilizing resources of Google App Engine. It is completely scalable and provides programming interfaces (APIs) for sharing the collected data with external applications and visualizing them on mobile devices.

II. Related Work

There is a great number of research works in pervasive healthcare sensors. Most of them deal with data management on the devices (e.g., using storage means like SD cards) or utilize intermediate nodes (e.g., mobile phones) or store the data directly on computer nodes. Only few works exist however that address the issue of data storage and management on the Cloud. Authors in [4] present a sensor-oriented cloud infrastructure. The presented platform is proprietary and the initial evaluation results are based on simulated sensors and do not include actual devices.

There is also available a number of Cloud-based services dedicated for storing sensor-based data. Pachube [19], Nimbits [22], ThingSpeak [20] and iDigi [21] are a few that could be mentioned.

Pachube has been one of the first on-line database service providers that allow developers to connect sensor data to the Web. It is a real-time data Cloud-based infrastructure platform for the Internet of Things (IoT) with a scalable infrastructure that enables users to build IoT products and services, and store, share and discover real-time sensor, energy and environment data from objects, devices & buildings around the world. The main features of the platform are: managing real time sensor and environment data, graphing and monitoring and controlling remote environments. In addition there is a great number of interfaces available for building sensor or mobile-based applications for managing the data on the Cloud infrastructure. One of the important features of Pachube that have facilitated its penetration as a IoT cloud service is that the basic usage if free, it is based on an open and easy accessible API and has a very interactive web site for managing sensor data.

Nimbits is a data processing service you can use to record and share sensor data on the cloud. It is a free, social and open source platform for the Internet of Things. With Nimbits, users can create data points on the cloud and feed changing numeric, text based, GPS, JSON or xml values into them. Data points can be configured to perform calculations, generate alerts, relay data to social networks and can be connected to SVG process control diagrams, spreadsheets, web sites and more. Nimbits offers a data compression mechanism, an alert management mechanism, and data calculation on the received sensor data using simple mathematic formulas.

ThingSpeak is another open source "Internet of Things" application and API to store and retrieve data from things using HTTP over the Internet or via a Local Area Network. With ThingSpeak, users can create sensor-logging applications, location tracking applications, and a social network of things with status updates. In addition to storing and retrieving numeric and alphanumeric data, the ThingSpeak API allows for numeric data processing such as time scaling, averaging, median, summing, and rounding. Each ThingSpeak Channel supports data entries of up to 8 data fields, latitude, longitude, elevation, and status. The channel feeds support JSON, XML, and CSV formats for integration into applications. The ThingSpeak application also features time zone management, read/write API key management and JavaScript-based charts.

iDigi Platform is a machine-to-machine (M2M) platform-as-a-service. iDigi Platform lowers the barriers to building secure, scalable, cost-effective solutions that seamlessly tie together enterprise applications and device assets. iDigi Platform manages the communication between enterprise applications and remote device assets, regardless of location or network. It makes connecting remote assets easy, providing all of the tools to connect, manage, store and move information across the near and far reaches of the enterprise. The platform includes the device connector software (called iDigi Dia) that simplifies remote device connectivity and integration. It allows the management (configure, upgrade, monitor, alarm, analyze) of products including ZigBee nodes. The application messaging engine enables broadcast and receipt notification for application-to-device interaction and confirmation. There are also cache and permanent storage options available for generation-based storage and on-demand access to historical device samples.

These services however, focus mostly on the visualization of the data and usually lack of secure data access and provision of interfaces for linkage to mobile or external applications for further processing.

The majority of the aforementioned works is based on proprietary architectures and communication schemes, which requires the deployment of specific software components. Furthermore, these works deal mostly with delivering data to healthcare applications and do not address issues of data management and interoperability issues introduced by the heterogeneous data resources found in modern electronic healthcare systems. The introduction of Cloud Computing infrastructure may provide data management and access functionality overcoming the aforementioned issues as discussed in previous lines. The concept of utilizing Cloud Computing in the context of pervasive healthcare information management is relatively new but is considered to have great potential [5], motivating us to deal with this important field.

III. The Proposed Architecture

This section describes the proposed architecture of the developed system, which is also illustrated in **Figure 2**. Wearable and body sensors collect data regarding the status of the patient, like heart rate, temperature, blood oxygen saturation, location, etc. the sensors have appropriate wireless connectivity or can communicate with other means to devices like a mobile phone. The latter are responsible for collecting the data acquired by the sensors and for forwarding them to the Cloud infrastructure. Simple and lightweight communication protocols (like Web Service calls) are utilized. Data exchange can be implemented over HTTP or HTTPS protocol for secure communication.

An appropriate cloud-based web application has been developed that manages the sensor data on Cloud resources. The

system exposes also various functions (such as alert management and data overview) on mobile devices or third applications (like a fall detection analysis) through Web Services.

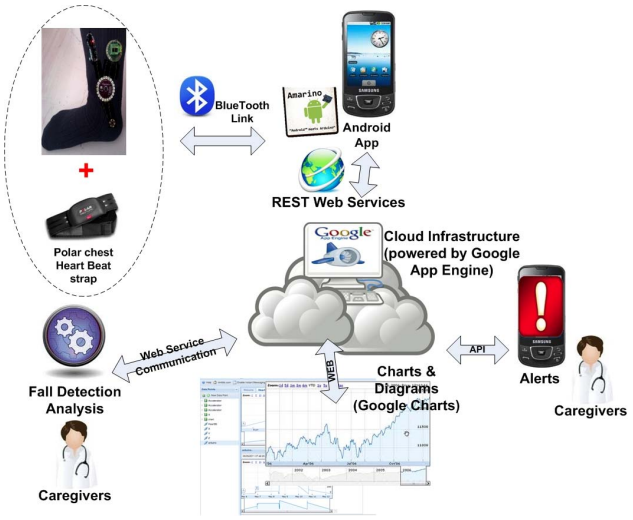

Figure 2. Illustration of the architecture, main components and interaction with users.

Currently, the most open and interoperable way to provide access to remote services and/or enable applications to communicate with each other is to utilize Web Services. The term Web Services is fairly self-explanatory, it refers to accessing services over the web. But, there's more to it than that, the current use of the term refers to the architecture, standards, technology and business models that make Web Services possible. According to various available definitions, Web Services are self-contained, self-describing, modular applications that can be published, located, and invoked across the Web. Web Services perform functions, which can be anything from simple requests to complicated business processes. In other words, Web Services are interoperable building blocks for constructing applications. A Web Service is usually identified by a URI (Unified Recourse Identifier).

A Web Service has WSDL (Web Service Description Language) definitions. These are computerized descriptions of what the Web Service can do, where it is located and how it can be used (referred as 'consumed') by the client application. To communicate with Web Services we need to use SOAP messages, which are XML based messages transported over Internet protocols like HTTP, SMTP, and FTP.

Web Services have certain advantages over other technologies:

- Web Services are platform-independent and language-independent, since they use standard XML languages. This means that my client program can be programmed in C++ and running under Windows, while the Web Service is programmed in Java and running under Linux.
- Most Web Services use HTTP for transmitting messages (such as the service request and response). This is a major advantage if you want to build an Internet-scale application, since most of the Internet's proxies and firewalls won't mess with HTTP traffic (unlike CORBA, which usually has trouble with firewalls).

In this work, we have placed all utilized sensors on a sock for better usability and wearability. The sensors are textile, can be easily sewed using conductive thread on the fabric and are even washable. More details on the materials and methods used are provided in the following section.

IV. The System in Practice: Modules and Initial Results

The platform consists mainly of two parts: the wearable part the collects and transmits motion and heartbeat data and the cloud infrastructure for storing the data. For the wearable part we have used textile accelerometers and a heartbeat chest strap by Polar ([1]). The latter sensors are connected to a textile version of the Arduino open hardware microcontroller platform ([6]), called LilyPad ([2]). Arduino is an open-source single-board microcontroller.

An Arduino board consists of an 8-bit Atmel AVR microcontroller with complementary components to facilitate programming and incorporation into other circuits. An important aspect of the Arduino is the standard way that connectors are exposed, allowing the CPU board to be connected to a variety of interchangeable add-on modules.

At a conceptual level, when using the Arduino software stack, all boards are programmed over an RS-232 serial connection, but the way this is implemented varies by hardware version. Serial Arduino boards contain a simple inverter circuit to convert between RS-232-level and TTL-level signals. Current Arduino boards are programmed via USB, implemented using USB-to-serial adapter chips such as the FTDI FT232. Some variants, such as the Arduino Mini and the unofficial Boarduino, use a detachable USB-to-serial adapter board or cable, Bluetooth or other methods.

The Arduino IDE is a cross-platform application written in Java, and is derived from the IDE for the Processing programming language and the Wiring project. It is designed to introduce programming to artists and other newcomers unfamiliar with software development. It includes a code editor with features such as syntax highlighting, brace matching, and automatic indentation, and is also capable of compiling and uploading programs to the board with a single click.

The LilyPad Arduino is a microcontroller board designed for wearables and e-textiles. It can be sewn to fabric and similarly mounted power supplies, sensors and actuators with conductive thread.

For the connection between the Polar monitor and the microcontroller, the Polar HeartRate Module has been utilized. Lilypad collects data through the appropriate embedded software and transmits them on an Android-based mobile phone through a Bluetooth interface. An appropriate application has developed for the Android that collects the data and forwards them to the Cloud. All textile sensors and the microcontroller have been sewed on a sock that can be worn easily by the user (see **Figure 3**).

A. Cloud Computing Utilization

Cloud Computing is a model for enabling convenient, on-demand network access to a shared group of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. Resources are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms (e.g., smart phones). Examples of resources include storage, processing, memory, network bandwidth, and virtual machines. Given the characteristics of Cloud Computing and the flexibility of the services that can be developed, a major benefit is the agility that improves with users being able to rapidly and inexpensively re-provision technological infrastructure resources. Device and location independence enable users to access systems using a web browser, regardless of their location or what device they are using (e.g., mobile phones). Multi-tenancy enables sharing of resources and costs across a large pool of users, thus allowing for centralization of infrastructure in locations with lower costs. Reliability improves through the use of multiple redundant sites, which makes Cloud Computing suitable for business continuity and disaster recovery. Security typically can be improved, due to centralization of data and increased availability of security-focused resources. Sustainability comes about through improved resource utilization, resulting in more efficient systems.

A number of Cloud Computing platforms are already available for pervasive management of user data, either free (e.g., iCloud [15], ¬Okeanos [18], and DropBox [17]) or commercial (e.g., GoGrid [14] and Amazon AWS [16]). Most of them, however, do not provide substantial developer support, to create custom applications and incorporate Cloud Computing functionality, apart from Amazon AWS. None of them is optimized for the provision of services to sensor-based applications.

B. The Cloud Infrastructure

For the Cloud infrastructure we have utilized the Google App Engine, as an open source way for hosting online applications on the Cloud. It is a cloud-computing platform for developing and hosting web applications in Google-managed data centers. Google App Engine makes it easy to build an application that runs reliably, even under heavy load and with large amounts of data. The main features of the Google App Engine include:

- Dynamic web serving, with full support for common web technologies
- Persistent storage with queries, sorting and transactions
- Automatic scaling and load balancing of the applications
- APIs for authenticating users and sending email using Google Accounts
- A fully featured local development environment that simulates Google App Engine on your computer
- Task queues for performing work outside of the scope of a web request
- Scheduled tasks for triggering events at specified times and regular intervals

App Engine's infrastructure removes many of the system administration and development challenges of building applications to scale to hundreds of requests per second and beyond. Google handles deploying code to a cluster, monitoring, failover, and launching application instances as necessary.

Google App Engine also provides all the appropriate application programming interfaces (APIs) for developing web applications that allow users to view visually the collected data and manage them. It also provides APIs for creating interoperable mobile applications and ways for integrating existing applications (e.g., a fall detection platform as in [3], [9] - [12]).

While other services let users install and configure nearly any *NIX compatible software, App Engine requires developers to use only its supported languages, APIs, and frameworks. Current APIs allow storing and retrieving data from a non-relational database; making HTTP requests; sending e-mail; manipulating images; and caching. Most existing Web applications can't run on App Engine without modification, because they require a relational database. Per-day and per-minute quotas restrict bandwidth and CPU use, number of requests served, number of concurrent requests, and calls to the various APIs, and individual requests are terminated if they take more than 30 seconds or return more than 10MB of data.

App Engine offers automatic scaling for web applications - as the number of requests increases for an application, App Engine automatically allocates more resources for the web application to handle the additional demand.

Google App Engine is free up to a certain level of consumed resources. Fees are charged for additional storage, bandwidth, or CPU cycles required by the application.

C. The Cloud Application

Google App Engine supports the hosting of applications developed either in Java, Python and the recently introduced Google's programming language called Go. We have selected Java to develop our application that receives sensor data, stores them in the Cloud infrastructure provided by Google and also provides a visualization web application and APIs for communicating with external applications and mobile devices. The web interface (see **Figure 4**) is a J2EE web application that allows user to create and manage data points related to sensor

data. The Google chart visualization API has been utilized in order to present diagrams with sensor data.

Sensor data are retrieved using a lightweight REST API that enables direct communication from the Arduino Lilypad microcontroller when direct wireless connectivity is available.

Here is an example HTTP POST to the Web application using the REST API:

```
POST /update HTTP/1.1
Host: cloudsensorsock.appspot.com
Connection: close
X-THINGSPEAKAPIKEY: (Write API Key)
Content-Type: application/x-www-form-urlencoded
Content-Length: (number of characters in message)


secret_key='key'&sensorX=value&sensorY=value…
```

The last part of the HTTP Post represents the actual content. It is composed by a secret key (utilized as an authentication mechanism) of 16bit long, and the sensor values to be stored into the database.

The data are stored using the App Engine's Datastore feature. The Datastore writes data in objects known as entities, and each entity has a key that identifies the entity. Entities can belong to the same entity group, which allows you to perform a single transaction with multiple entities. Entity groups have a parent key that identifies the entire entity group. App Engine's infrastructure takes care of all of the distribution, replication and load balancing of data.

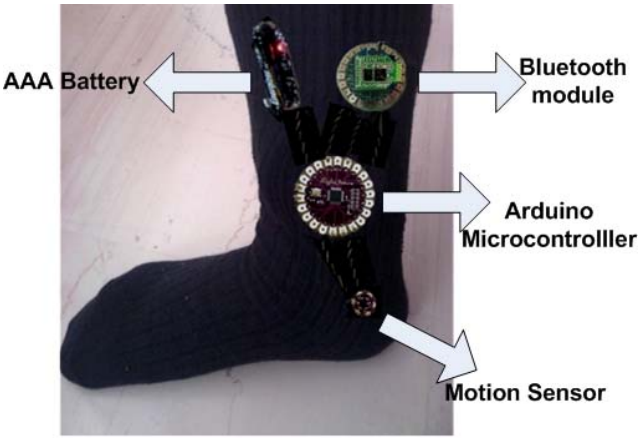

Figure 3. The CloudSensorSock and the main hardware modules as sewed on the final prototype.

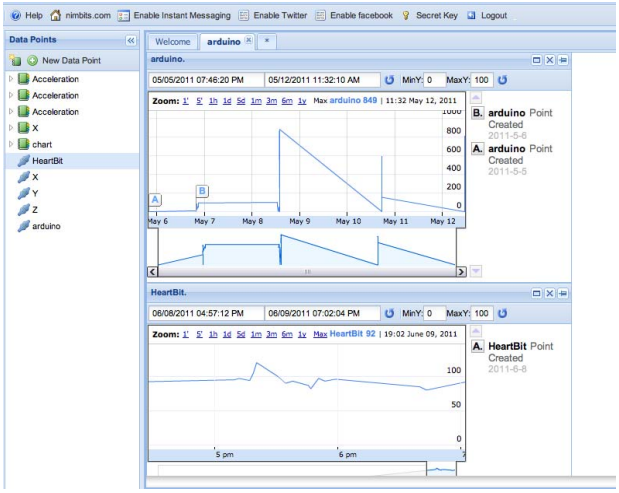

Figure 4.The web interface hosted on Cloud for viewing sensor data.

D. *Initial Evaluation Results*

During the initial experimentation with the system, a drop packet rate of 20-30% has been detected. This fact is either due to the Arduino low resources for high rate sampling of sensors and transmitting the data at the same time, or due to network congestion because of the repetitive REST calls at such a high sampling rate (i.e. 10 acceleration samples per second). In order to address this issue, a memory buffer has been introduced on the Arduino side that collects motion data during a 10 second time frame and then transmits the latter to the Cloud. This way the drop rate has been minimized between 2-5%, which is quite acceptable for the application.

V. CONCLUSIONS

The proposed solution is lightweight, non-invasive and low-cost (under 200$) and stores all data on the Cloud. Based on preliminary results the battery can last for more than 24hrs continuously monitoring and transmitting data regarding user's movement and heartbeat rate. Users can visually view acquired data in charts using developed web applications and receive alerts on mobile phones. Data can also be utilized by fall detection platforms and be used for emergency response and treatment.

The presented system is unique as a dedicated solution for managing patient-related data on the cloud and that utilizes both open hardware and open software resources for developing the hardware and software parts of the platform. It allows direct communication of the sensor devices with the Cloud application due to the lightweight API used, while it is highly scalable in the context of data stored, users and sensors supported.

Open issues that need to be addressed are the security of privacy of data and the energy efficiency of the textile sensors and microcontroller platform, in order to extend the system autonomy.

ACKNOWLEDGMENT

This work was partially supported by Information Society Technology program of the European Commission "e-Laboratory for Interdisciplinary Collaborative Research in Data Mining and Data-Intensive Sciences (e-LICO)" (IST-2007.4.4-231519).

REFERENCES

[1] Polar, www.polar.com

[2] The Arduino LilyPad, http://arduino.cc/en/Main/ArduinoBoardLilyPad

[3] Charalampos Doukas, Ilias Maglogiannis, "Emergency Fall Incidents Detection in Assisted Living Environments Utilizing Motion, Sound and Visual Perceptual Components", in IEEE Transactions on Information Technology in Biomedicine, vol. 15, no. 2, pp. 277 – 289, March 2011.

[4] Yuriyama, M.; Kushida, T.; , "Sensor-Cloud Infrastructure - Physical Sensor Management with Virtualized Sensors on Cloud Computing," *Network-Based Information Systems (NBiS), 2010 13th International Conference on* , vol., no., pp.1-8, 14-16 Sept. 2010.

[5] Ofer Shimrat.: Cloud Computing and Healthcare. San Diego Physician, pp. 26-29 (2009)

[6] The Arduino Open Source microcontroller platform, http://www.arduino.cc

[7] Wearable microcontroller solution, LilyPad Arduino, http://arduino.cc/en/Main/ArduinoBoardLilyPad

[8] The Android Open Accessory Development Kit, http://developer.android.com/guide/topics/usb/adk.html

[9] Charalampos Doukas, Ilias Maglogiannis.: An Assistive Environment for Improving Human Safety Utilizing Advanced Sound and Motion Data Classification. Accepted for publication in Universal Access in the Information Society, Springer.

[10] Charalampos Doukas, Ilias Maglogiannis.: Advanced Classification and Rules-Based Evaluation of Motion, Visual and Biosignal Data for Patient Fall Incident Detection. Artificial Intelligence Techniques for Pervasive Computing, International Journal on AI Tools (IJAIT), World Scientific Press, vol. 19, issue 2, pp. 175-191 (2010)

[11] Charalampos Doukas, Ilias Maglogiannis.: Advanced Patient or Elder Fall Detection based on Movement and Sound Data. 2nd International Conference on Pervasive Computing Technologies for Healthcare 2008.

[12] Charalampos Doukas, Ilias Maglogiannis, Philippos Tragkas, Dimitris Liapis, Gregory Yovanof.: Patient Fall Detection using Support Vector Machines. In Proc. of the 4th IFIP Conference on Artificial Intelligence Applications & Innovations (AIAI), Sept. 19-21, Athens, Greece (2007)

[13] George Reese, Cloud Application Architectures: Building Applications and Infrastructure in the Cloud, O'Reilly Media, Paperback (April 17, 2009), ISBN: 0596156367.

[14] GoGrid Storage Services, http://www.gogrid.com

[15] iCloud, http://www.icloud

[16] Amazon Web Services (AWS), http://aws.amazon.com/

[17] DropBox, https://www.dropbox.com

[18] ~Okeanos cloud services for the Greek academic community, http://okeanos.grnet.gr

[19] The Pachube Feed Cloud Service, http://www.pachube.com

[20] Internet of Things – ThingSpeak service, http://www.thingspeak.com

[21] iDigi Device Cloud, http://www.idigi.com

[22] Nimbits Data Logging Cloud Sever, http://www.nimbits.com