PUC

ISSN 0103-9741

Monografias em Ciência da Computação
n° 04/2014

The Mobile Hub: Enabling applications for the Internet of Mobile Things

Luis Talavera

Markus Endler

Igor Vasconcelos

Rafael Vasconcelos

Marcio Cunha

Francisco Silva da Silva

Departamento de Informática

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO

RUA MARQUÊS DE SÃO VICENTE, 225 - CEP 22453-900

RIO DE JANEIRO - BRASIL

Monografias em Ciência da Computação, No. 04/2014 ISSN: 0103-9741
Editor: Prof. Carlos José Pereira de Lucena October, 2014

The Mobile Hub Concept: Enabling applications for the Internet of Mobile Things[1]

L.E. Talavera, M. Endler, I. Vasconcelos,
R. Vasconcelos, M. Cunha

Francisco Silva da Silva[1]

[1] Department of Informatics - Federal University of Maranho

{lrios,endler,ivasconcelos,rvasconcelos,mcunha}@inf.puc-rio.br
fssilva@deinf.ufma.br

Abstract. Few studies have investigated and proposed a middleware solution for the Internet of Mobile Things (IoMT), where the smart things (Smart Objects) can be moved, or else can move autonomously, and yet remain accessible and controllable remotely from any other computer over the Internet. Examples of mobile Smart Objects include vehicles of any nature, wearable devices, smart watches, sensor tags, mobile robots, Unmanned Aerial Vehicles (UAVs), i.e., any mobile thing with embedded sensors and/or actuators. In this context of general and unrestricted mobility of Smart Objects, the main challenge is to ensure endured connectivity and discovery of Smart Objects, as well as the efficient scalable and reliable remote access to its sensors and actuators. This paper describes the concept a Mobile Hub as a key enabler of the Internet of Mobile Things, its design and an initial implementation of the concept for Android and a single WPAN technology: Bluetooth Low Energy. The M-Hub is the natural extension of the Scalable Data Distribution Layer (SDDL), a mobile communication middleware developed by our group that adopts a mobile-cloud architecture and provides scalable mobile-mobile communication and processing capabilities. Preliminary experiments have shown that our implementation for BLE delivers good mobility responsiveness and that the concept is suitable for applications which have to deal with the mobility of Smart Objects/ things.

Keywords: Context-Awareness, middleware, context modeling, mobile computing

[1] ContextNet project is being supported by TecGraf Institute (PUC-Rio), by the *PUC-Rio Microsoft Open Source Alliance* and by several CNPq stipendia.

Resumo. Poucos estudos têm investigado e proposto soluções de middleware para a Internet das Coisas Móveis (IoMT), onde as coisas inteligentes (Smart Objects) podem ser movidas, ou mover-se de forma autânoma, e ainda permanecer acessível e controlável remotamente a partir de qualquer outro computador da Internet. Exemplos de objetos móveis inteligentes incluem veículos de qualquer natureza, dispositivos portáteis, relógios inteligentes, etiquetas de sensores, robôs móveis, veículos aéreos não tripulados (UAVs), ou seja, qualquer coisa móvel com sensores e / ou atuadores embarcados. Neste contexto, de mobilidade geral e irrestrita de objetos inteligentes, o principal desafio é garantir e manter a conectividade e descoberta de objetos inteligentes, bem como o acesso remoto escalável e confiável eficiente de seus sensores e atuadores. Este artigo descreve o conceito de um Mobile Hub (M-Hub), como um fator-chave da Internet das Coisas Móveis, seu design e uma implementação inicial do conceito para Android e uma única tecnologia WPAN: Bluetooth Low Energy. O M-Hub é a extensão natural do Scalable Data Distribution Layer (SDDL), um middleware de comunicação móvel desenvolvido pelo nosso grupo que adota uma arquitetura móvel em nuvem e fornece capacidades de comunicação e de processamento escalável de clientes móveis para móveis. Experimentos preliminares mostraram que a nossa aplicação para BLE oferece boa mobilidade de resposta e que o conceito é adequado para aplicações que têm de lidar com a mobilidade de objetos inteligentes / coisas.

Palavras-chave: Percepção de contexto, middleware, modelagem de contexto, computação móvel

In charge for publications:
Rosane Teles Lins Castilho
Assessoria de Biblioteca, Documentação e Informação
PUC-Rio Departamento de Informática
Rua Marquês de São Vicente, 225 - Gávea
22453-900 Rio de Janeiro RJ Brasil
Tel. +55 21 3114-1516 Fax: +55 21 3114-1530
E-mail: bib-di@inf.puc-rio.br
Web site: http://bib-di.inf.puc-rio.br/techreports/

1 Introduction

Despite the huge number of potential applications and the increasing proliferation of appliances with embedded processing and wireless communication capacity, yet there is no widely accepted approach, established standards and consolidated technologies for the Internet of Things at global scale. In other words, Internet-wide communication and processing of data from tens of billions of sensors and actuators within devices and smart objects is still a challenge. In particular, very few studies have focused on the Internet of Mobile Things (IoMT), in which the connectable things (or Objects) can be moved or can move independently, and yet remain remotely accessible and controllable from anywhere in the Internet. Mobile Objects (M-OBJs) may have very different size, purpose and complexity - they may span from from terrestrial vehicles of any type (cars, busses, etc.), over mobile domestic or industrial robots, aerial robots (UAVs), to very tiny and light-weight wearable devices, badges or tokens, and eventually, even "smart dust". In fact, a M-OBJ may be any movable object that carries sensors and/or actuators and has provides some means of wireless connectivity. In this context of general and unrestricted mobility of Smart Objects, the main challenge is to ensure best possible connectivity with mobile sensors/actuators, fast discovery, seamless handovers, and continuous tracking and access to the M-OBJ's resources, capability-based search and selection, as well as management of data streams to and from the M-OBJs in efficient, scalable ways.

Several wireless technologies for short range and low power connectivity (the last 100 meter connectivity - last100m) have recently emerged, such as Bluetooth 4.0, ZigBee, NFC, WiFi Direct, 6lowPAN, or ANT+, which support some interconnection among Smart Objects. However, most of these wireless technologies have the common drawback of supporting connectivity only within limited scope, and imposing significant latency on the inter-device discovery and pairing procedures, which on the one hand prevents the M-OBJ's access from anywhere on the globe, and on the other hand hinders fast reconnections and re-arrangements in stringent mobility scenarios. Hence, in this project we aim to investigate several research challenges and issues associated with the Internet of Mobile Things (IoMT), where M-OBJs of any sort are accessed through the now most disseminated pervasive devices: smart phones and other portable and personal devices such as tablets.

According to Francis daCosta [3], IPv6 does not solve all IoT problems because management, rather than addressing and routing, will be the biggest challenge of IoT. In fact, IP-based protocols will neither be supported by the vast majority of Smart Objects, nor will their over-provisioned and reliable services be suited to most IoT applications. The reason is that IP-based protocols *"...are intrinsically designed for high-duty cycles, large data streams, and reliability."*, while in IoT communication involves small but frequent messages, where each message individually is unimportant, but the statistical properties of the corresponding data flows carry the relevant pieces of information. Moreover, the IoT networking will not have a flat Peer-to-Peer architecture as some believe, since *"...many devices at the edge of the network have no need to be connected with other devices at the edge of the network..."*, and since *"...the communications intelligence and functionality does not exist within the end devices, other devices - propagator nodes - must be present in the network to transport data efficiently and manage the data flows..."* [3].

Thus, considering that smart phones are becoming ubiquitous[2], cheeper and more

[2]According to eMarketer, smartphone users will total 1.75 billion users in 2014 - almost 24% of the

powerful, and that disconnections and message loss will be the norm in IoT, where reliable delivery of single messages from/to M-OBJs will be less important, smart phones and tablets are the natural candidates for serving as IoT propagator nodes. This makes us propose the concept of *Mobile Hubs(M-Hub)* as the intermediate between the myriad of different Smart M-OBJs and the long-haul Internet connection.

By supporting the Internet of Mobile Thing (IoMT) through this concept of M-Hub we are, in fact, coping with a more general - and harder - problem than traditionally addressed by the IoT community, that usually assumes that both the peripheral/edge devices (the Objects) and the propagator nodes are fixed at some place. Essentially, the IoMT paradigm considers any situation in which the *relative position and velocity* between the M-OBJs and the M-Hub is variable and may change anytime, and where M-OBJs may be reached through different and even multiple M-Hubs over time. Therefore, IoMT encompasses situations where:

1. the M-OBJs are permanently associated with a place and the M-Hub moves across the places to opportunistically interact with the M-OBJs, e.g. for sampling local sensor data;

2. the M-OBJs are attached to movable items, while the M-Hub is linked to a place, e.g. a warehouse, and the M-OBJs reveal their presence to the M-Hub whenever they get close to it and

3. the M-OBJ stays in co-movement with the M-Hub for a certain period of time, e.g. passengers in a vehicle, health-sensors carried by a patient, etc.

Of course, this more generic model of IoMT brings along the burden of potentially much more indeterminism in the form of unpredictable sensor/actuator availability, less reliability, more connectivity volatility, higher probability of interferences, and much more. Nevertheless, we believe that there exist several (non-critical) applications which have this intrinsic characteristics of unrestricted mobility and which may benefit from our approach. In particular, we envision several applications that will benefit from another important characteristics of the M-Hub: its ability to enrich the M-OBJ's data streams with contextual information, obtained from its own sensors, such as its current geographic position. This feature will open up to applications new ways of classifying, filtering or searching for data to/from the M-OBJs. Finally, it is worth noting that in IoMT mobility is just an option: M-OBJs and M-Hubs may also be deployed in a static configuration, and be associated with a specific place, for applications such as Smart Home/Buildings, etc.

The remainder of this paper is structured as follows. In the next section we describe two potential applications for IoMT, and in Section 3 give an overview of the ContextNet project and its already implemented mobile communication infrastructure. Then, in section 5, we present the concept and main functions of the Mobile Hub, as well as its general architecture and its main components. In Section 6 we describe some details our current M-Hub prototype and results of performance tests done so far. In Section 7 related work is discussed, and 8 contains a discussion of other roles that the M-Hub may assume in future applications. Finally, in Section 9, we draw concluding remarks and point to future work.

world population!

2 Possible IoMT applications

In the following we present two hypothetic IoMT applications, which require the concept of a Mobile Hub, and which illustrate the three situations presented in Section 1.

Application 1: In a region with high density of smartphone users, such as a metropolitan area, we can imagine the need for collaborative air quality monitoring. Common citizens may obtain tax incentives to install and deploy affordable Wireless Air Monitoring Stations (WAMS) in their yards, along neighborhood driveways, parks or other public spaces. All such WAMS would have CO, NO_2, SO_2, Lead sensors, etc., a short range and low-power wireless interface, be weather-resistant and run on solar energy. In this context, the collaborative monitoring app would be a crowd-sourced one where pedestrians passing close to some WAMS would donate their smartphone's Internet connectivity and energy to upload the current collected sensor data from the nearby WAMS to a city-wide monitoring service in the cloud, where all this information would be presented on a map both on-line and in consolidated statistics, for access by any citizens. Actually, through this IoMT application air quality indeed would not be be monitored uniformly for the entire city region, but only at those parts of the city where it is most relevant, i.e., the places with much intense pedestrian and bike traffic. Moreover, some citizens may even opt for a air monitoring on the go-attitude, carrying a smaller and lighter version of the WAMS on their bike baskets or their knapsacks, so as to measure the air pollution on their ways. And through their smartphone, this data would be uploaded to the central monitoring service.

Application 2: Nowadays, several goods and merchandise require specific minimal transportation and storage conditions on their routes from producer to consumer. For example, meat and some fruits need ambient temperatures of less than 10 degrees Celsius, special flowers and plants should be in ambients with air humidity above certain level, livestock requires smooth movement as well as places with sufficient air circulation (i.e., O_2 concentration), etc. Hence, it would be advantageous to be able to monitor the ambient and movement conditions along all the transport path of these and other "sensitive" merchandise. This could be done by placing some smart sensors (M-OBJs) close to the goods, and having the sensor values probed in all the stages of transportation and intermediate storage. These data could then be send in real-time to interested users (e.g. the customer, or the transportation company) so as to early detect some non-conformance in the transportation conditions, or else, be the input for transport reports. Such a monitoring could be achieved by having M-Hubs placed in the array of vehicles (trucks), vessels or delivery personal involved in the transportation. By such, as soon as the merchandise cargo is loaded, stored or picked by the next transportation means, the M-Hub would connect with the M-OBJs, send messages acknowledging the arrival/ handing-over of the goods to the clients, and start receiving, checking the M-OBJs sensed data, and eventually sending alert notifications about inadequate transport conditions.

As can be noticed, Application 1 illustrates situations 2 and 3, whereas Application 2 illustrates situation 1.

3 The ContextNet project

The goal of the ContextNet project is to extend the SDDL communication middleware [11, 14, 15], its protocols, APIs and services to enable discovery, management and access to Mobile Objects (M-OBJ) through Mobile Hubs (M-Hubs). While the former can be very simple sensor or actuator devices with no significant processing and storage capacity (dump objects), the latter are resource-full portable personal devices (smart phones or tablets) that will "bridge the gap" between the Internet connection with the SDDL Core (e.g. executing in a cloud) and the short-range wireless connections established with nearby M-OBJs. For this short-range wireless connectivity, we have designed S2PA, a general and technology-independent communication protocol and API which runs on the M-Hub. As its first realization we are implementing S2PA it for Bluetooth 4.0 (*Bluetooth Smart* or *Bluetooth Low Energy - BLE*).

In fact, BLE is emerging as a very promising technology for WPAN, because it is power efficient, enables fast discovery of devices and supports approx. 2500 simultaneous connections. But the most important reason for choosing BLE is the fact that being made available on a growing array of Android, iOS, Blackberry smart phones and moreover is being deployed into many peripheral devices, gadgets, beacons and small Sensor Tags[3].

Although the ContextNet IoMT support encompasses some other services on the SDDL core, presented in [13], in this paper we will explain the design and preliminary performance results of our M-Hub/S2PA prototype implementation for Bluetooth LE.

4 SDDL Middleware: an Overview

The Scalable Data Distribution Layer (SDDL) is a communication middleware that connects stationary DDS nodes in a wired core network to mobile nodes with an IP-based wireless data connection. Some of the stationary nodes are information and context data processing nodes, others are gateways for communication with the mobile nodes, and yet others are monitoring and control nodes operated by humans, and capable of displaying the mobile nodes current position (or any other context information), managing groups, and sending message to the mobile nodes[4]. SDDL employs two communication protocols: the Data Distribution Service (DDS) Real Time Publish/Subscribe Protocol for the wired communication within the SDDL core, and the Mobile Reliable UDP (MR-UDP) [12] for the inbound and outbound communication between the core network and the mobile nodes. DDS [9] is a standard of the OMG that specifies a peer-to-peer middleware architecture for real time and high-performance data distribution, with Quality of Service (QoS) contracts between producers and consumers of data (e.g., reliable communication, data persistency, priority lanes, etc.). In a nutshell, MR-UDP is Reliable-UDP with mechanisms for tolerating intermittent connectivity, dynamic IP address changes of the Mobile nodes and reaching these nodes behind firewalls/NATs. It is used by the mobile nodes to connect with a special type of SDDL Core node called *Gateway (GW)*, of which any number can be deployed in the SDDL Core. Each Gateway maintains one independent MR-UDP connection with each mobile node, and is responsible for translating application-messages from MR-UDP to the intra-SDDL core protocol, and, in the opposite direction, converting core

[3]See: www.bluetooth.com/Pages/Bluetooth-Smart-Devices-List.aspx

[4]A mobile node is the generic term for an end user terminal or a Mobile Hub

messages to MR-UDP messages and delivering them reliably to the corresponding mobile nodes.

The SDDL Core has several other specialized nodes: the PoA-Manager, which holds the addresses list of all currently active Gateways and eventually requests mobile nodes to switch Gateways; GroupDefiners evaluate the group-memberships of all mobile nodes (based on some attribute of their inbound application messages) and manage the reliable delivery of group-cast messages to the mobiles, and the Controller, which provides a Map-based user interface to visualize and interact with any mobile node of the system. Finally, there is the *ClientLib*, a Java/Lua library which establishes and manages a MR-UDP connection of a mobile node client and the Gateways. It hides most MR-UDP details and message retransmission issues from the application layer, and also supports a fully application-transparent handover of the mobile node between SDDL Gateways.

More information about SDDL components and applications can be found in the papers [11, 15]. The interested reader can download a VM with pre-installed SDDL, as well as find examples and tutorials for implementing SDDL-based applications in Java, Android and Lua[5].

5 The Mobile Hub concept and its design

Since the Mobile Hub plays a crucial role as the intermediary between the Mobile Objects and the Internet, it must have full-fledged processing power, sufficient memory, and network interfaces both to Mobile Internet (2G/3G/4G) and to some low-range, low-power WPAN communication technology. We further assume that the M-Hub has some means of learning its current location, e.g. by sampling its GPS sensor, through network-based positioning, or else, by manual configuration (when used in stationary settings).

The main mandatory tasks of the M-Hub thus include:

Discovery, monitoring and registration of nearby M-OBJs periodically, the M-Hub will scan for nearby M-OBJs announcing their IDs and capabilities. This information about reachable M-OBJs will be kept stored in the M-Hub database and eventually forwarded to some service executing in the SDDL core.

Connecting to a M-OBJ depending on the kind of interaction (and the WPAN technology capabilities) a stable communication link may be established with some M-OBJ, over which the M-Hub will execute a request-reply protocol.

Protocol Transcoding Data packets and messages from/to M-OBJs may have different formats and encodings. Thus, the M-Hub will transcode them from/to serialized objects, encoded with Protocol Buffers, and transmitted over the MR-UDP connection.

Caching of recent M-OBJ sensor/status information in order to optimize communication over the Mobile Internet, the M-Hub may group several pieces of sensor data or commands from the set of nearby M-OBJs into a single "bulk message" for transmission. And in order to do this it will cache the most recent (the current) data items obtained from the M-OBJs.

[5]http://www.lac-rio.com/dokuwiki/doku.php?id=tutorial

Sending requests to M-OBJs depending on the services/ resources offered by the M-OBJ, the M-Hub will periodically or sporadically send queries about sensor readings and/or the M-OBJ's current state, or commands to set some parameter or drive some local actuator of the M-OBJ, e.g. rotate 45 degrees left;

Managing handover of M-OBJs to/from other M-Hubs since moving M-OBJs may enter and leave the WPAN coverage area of different M-Hubs over time, but one should attempt to keep the M-OBJs as long as possible accessible through any M-Hub, it is necessary that M-Hubs also keep track of other, nearby M-Hubs that may take over the role of serving a M-OBJ that is supposedly "drifting away".

In order to perform these functions, the M-Hub will use the ClientLib for communication with the Gateways, and the S2PA API for discovering and interacting with the M-OBJs. In our current M-Hub prototype, we have implemented S2PA only for Bluetooth Smart technology.

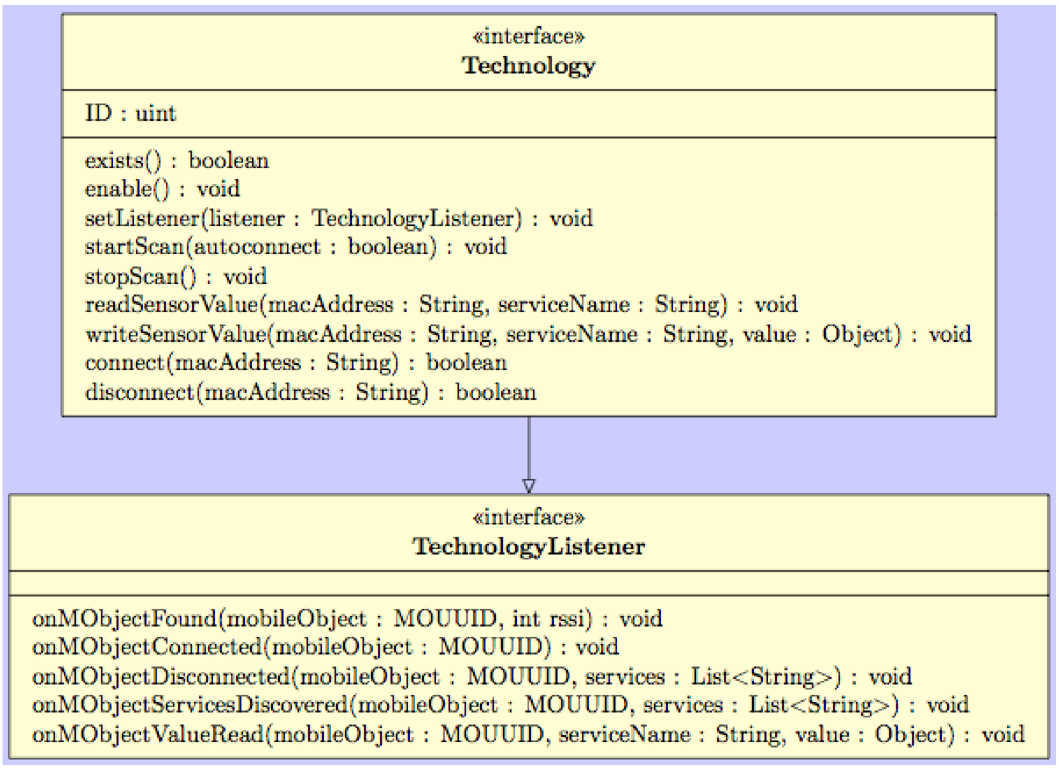
5.1 Short-Range Sensor, Presence and Actuation API

The *Short-Range Sensor, Presence and Actuation (S2PA)* API was designed to be a generic interface for short-range communication between the M-Hub and M-Objects, that can be directly mapped to the specific capabilities of the underlying short-range wireless communication technologies (a.k.a., WPAN). To achieve this we identified some basic functions that most of these technologies should implement: 1) Discovery and connection of M-Objects, 2) Discovery of services provided by M-Objects, 3) Read and write of service attributes (e.g., sensor values, and actuator commands) and 4) Notifications about disconnection of M-Objects. Hence, each WPAN should implement the following interface that defines these features in order to be used.

As one of its main elements S2PA defines the *Technology Interface*, as shown in Figure 1. The Technology interface posses a unique ID that is defined at programming time to identify each technology (e.g. BLE, ANT+, etc). Also, we believe that the interface has the required methods to handle different short-range protocols that are based on the idea of things. Some functions that need to be explained are the followings: **exists,** verifies if the technology exists on the device, **readSensorValue,** and **writeSensorValue,** request a read or a write of a sensor respectively, where serviceName represents the sensor and could be "Temperature", "Humidity", etc. All the important information that a technology can offer is sent to the TechnologyListener, where is the main logic of the S2PA Service, since it takes care of sending the information from the different technologies to the Connection Service.

5.2 Main Components

The M-Hub will be multi-threaded and will consists of the following four local services and two managers, all executing in background. The **LocationService** is responsible for sampling the M-Hub's current position and attaching it to whatever message is sent by the M-Hub to the Gateway (GW). The **S2PA Service** uses the S2PA library to interact with all nearby M-OBJs that "talk" the WPAN communication technologies supported by S2PA. This service is responsible for periodically doing scanDevices in all the supported WPAN technologies, registering discovered the M-OBJ's IDs and their capabilities in the



«interface»
Technology
ID : uint
exists() : boolean
enable() : void
setListener(listener : TechnologyListener) : void
startScan(autoconnect : boolean) : void
stopScan() : void
readSensorValue(macAddress : String, serviceName : String) : void
writeSensorValue(macAddress : String, serviceName : String, value : Object) : void
connect(macAddress : String) : boolean
disconnect(macAddress : String) : boolean
«interface»
TechnologyListener
onMObjectFound(mobileObject : MOUUID, int rssi) : void
onMObjectConnected(mobileObject : MOUUID) : void
onMObjectDisconnected(mobileObject : MOUUID, services : List<String>) : void
onMObjectServicesDiscovered(mobileObject : MOUUID, services : List<String>) : void
onMObjectValueRead(mobileObject : MOUUID, serviceName : String, value : Object) : void

Figure 1: Main two interfaces of the S2PA

SensService Registry, and transcoding sensor data and commands from the specific *SF* protocol format to Java objects to be transmitted to the GW, and vice-versa. Mobile Internet messages are received from and sent to the GW by the **ConnectionService**, which runs the ClientLib and manages a *msg buffer* of ready-to-send messages. These messages are either created by other mobile apps running on the device, or assembled by the S2PA Service. It is also important to mention that some messages (like connection or disconnection) won't be added to the buffer, instead they will be sent immediately. These messages are either created by other mobile apps running on the device, or assembled by the S2PA Service. The periodicity and duration of all of these three services' actions, is influenced by device's current energy level (LOW, MED, HIGH) which in turn will be controlled by the **Energy Manager**, which from time to time sample's the device's battery level and checks if it is connected to a power source. Finally, the **Handover Manager**, queries some specific data produced by the S2PA Service, such as the sensed RF signal strengths of nearby devices, and according to the situation, interacts with other M-Hubs (detected nearby) so as to proactively share information and parameters about M-OBJs, and ultimately swap responsibility for handing-out or handling-in M-OBJs, with these M-Hubs.

Figure 4 shows more details of the interactions between these components of M-Hub. *SensorTag, DevType2*, etc.. are modules that handle the information received from/ sent to specific M-OBJs. Each of them extends the `TechnologyDeviceService` class, and

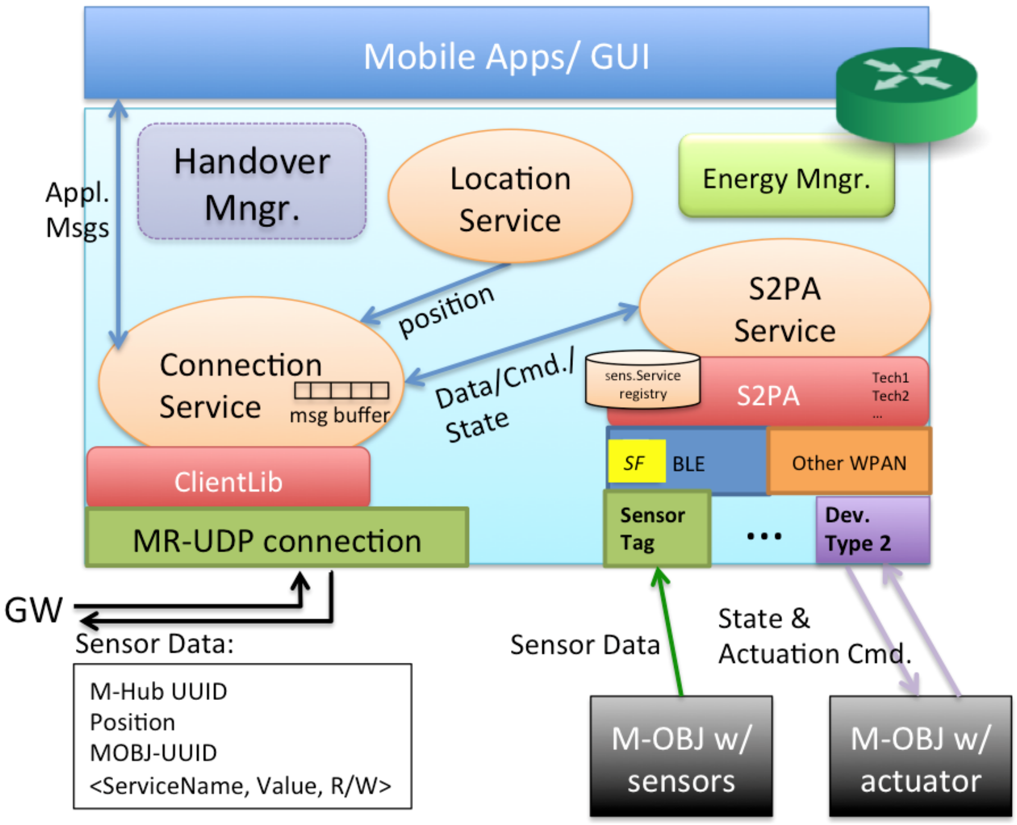


Figure 2: M-Hub's main components while it interacts with two M-OBJs, with different WPAN technologies.

overrides its abstract methods, among which the most important are the `convert()`, which handles the transformation of the raw data (bytes array) received from the M-OBJ's sensors to a JSON array format. The *SF* module performs the aforementioned conversion and is specific for each type of M-OBJ.

The S2PA Service is thus capable of managing several short-range communication technologies by calling generic methods (and registering listeners), that are mapped to the classes and methods of the different WPAN modules. As soon as a new M-OBJ is detected, S2PA Service places its services into the *Sensor Service Registry* , and starts getting data from the M-OBJ, which are then handed over to the *Connection Service*, where they are stored in a buffer (*msg buffer*) . The ConnectionService may also interact with any mobile app executing on the smartphone (the upper tier), forwarding application messages between the *ClientLib* and the apps, in both directions (see vertical arrow at the left). The ConnectionService periodically also receives position data from the *Location Service*, which is then simply added any ClientLib outbound message. The Energy Manager is the one that controls the periodicity intervals for all the three Services, depending on the current battery level, and if the M-Hub is plugged or not to the power supply. For example, if energy supply is *LOW*, *MEDIUM* or *HIGH* the S2PA Service will perform the scan (for all M-OBJs on all WPAN technologies) every 20, 30 and 40 seconds, respectively. These thresholds are configurable for different smartphone devices.

6 Current Status and Preliminary Results

Our current M-Hub prototype is for the Android platform and supports only a single WPAN, namely Bluetooth LE. It extends a previous set of Android components called ELISA (Energy-aware cLIent Service Library for Android)[6] by adding the *S2PA Service* and the modules for Bluetooth LE. In this prototype, we have not yet implemented the Handover Manager.

Nevertheless, our prototype already implements the first four mandatory functionalities (c.f., Section 5) of the M-Hub. It discovers, connects and starts receiving sensor data updates from any number of nearby M-OBJs. In our case, we tested it with off-the-shelf SensorTags[7]. This SensorTag has 6 sensors and no actuator.

At the level of S2PA Service it is only relevant to get all the sensor data from each M-OBJ. For this, we could either use the function `readSensorValue()` of the `Technology` Interface (see Figure 1) or take advantage of the special feature of Bluetooth LE, which allows us to receive a sensor data as soon as it changes, in the listener function `onMObjectValueRead()`.

The SensorTag's sensor data is then buffered in the *Connection Service* and sent periodically to a monitoring node in the SDDL Core. This monitor displays the M-Hubs on a map, together with the most recent sensor data obtained from the M-OBjs in the M-Hub's vicinity. Figure 3 shows two M-Hubs and the M-OBjs sensor readings.

Figure 5 show two screens of an Android app for visualizing the M-Hub settings and received values: i.e., the connection, location and scan settings, (left screen) as well as the discovered M-OBJs, their MO-UUIDs, and the current values obtained from each of its sensors (right screen).

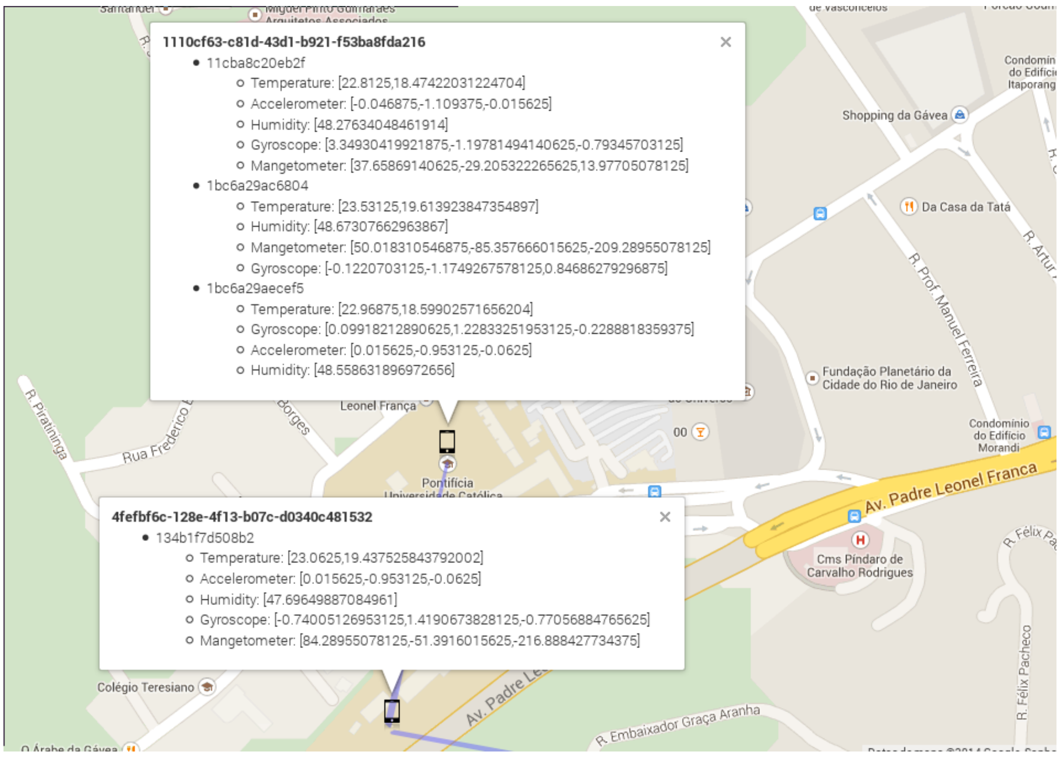
6.1 Preliminary Performance Tests

Using this prototype, we already did some preliminary experiments to access the latency of the discovery & connection-to process of the M-Hub with up to four Sensor Tags, and obtained encouraging results. For these experiments, we configured S2PA to perform a WPAN scan every 3 seconds and let the scan last for 2 seconds. The smartphone used as M-Hub for the experiments, is a Motorola Moto X (2013) with Android 4.4.2 KitKat. The notebook used to run the SDDL Core (Gateway and Controller/Monitor) for all the experiments is an ASUS Intel(R) Core(TM) i7-4500U CPU 1.80GHz with 5857 MB of RAM, running Arch Linux (Kernel 3.16.3-1-ARCH). The type of WLAN used is IEEE 802.11bgn.

In our first tests, we measured the Connection Time (CoT), Services Discovered Time (SDT), and Enable Notification Time (ENT), all in seconds, in Bluetooth LE for a single SensorTag, both for the first connection of the M-Hub with the M-OBJ (Table 1), and then for subsequent M-Hub to M-OBJ connections (Table 2). We ran each experiment 12 times and calculated the mean value and the standard deviation.

[6]ELISA - http://www.lac-rio.com/software/elisa-energy-aware-client-services-library-android

[7]Texas Instruments CC2541 Sensor Tag - http://www.ti.com/lit/ml/swru324b/swru324b.pdf



```
1110cf63-c81d-43d1-b921-f53ba8fda216
  • 11cba8c20eb2f
      o Temperature: [22.8125,18.47422031224704]
      o Accelerometer: [-0.046875,-1.109375,-0.015625]
      o Humidity: [48.27634048461914]
      o Gyroscope: [3.34930419921875,-1.19781494140625,-0.79345703125]
      o Mangetometer: [37.65869140625,-29.205322265625,13.97705078125]
  • 1bc6a29ac6804
      o Temperature: [23.53125,19.613923847354897]
      o Humidity: [48.67307662963867]
      o Mangetometer: [50.018310546875,-85.357666015625,-209.28955078125]
      o Gyroscope: [-0.1220703125,-1.1749267578125,0.8468627929 6875]
  • 1bc6a29aecef5
      o Temperature: [22.96875,18.59902571656204]
      o Gyroscope: [0.09918212890625,1.22833251953125,-0.2288818359375]
      o Accelerometer: [0.015625,-0.953125,-0.0625]
      o Humidity: [48.558631896972656]

4fefbf6c-128e-4f13-b07c-d0340c481532
  • 134b1f7d508b2
      o Temperature: [23.0625,19.437525843792002]
      o Accelerometer: [0.015625,-0.953125,-0.0625]
      o Humidity: [47.69649887084961]
      o Gyroscope: [-0.74005126953125,1.4190673828125,-0.77056884765625]
      o Mangetometer: [84.28955078125,-51.3916015625,-216.888427734375]
```

Shopping da Gávea
Da Casa da Tatá
R. Prof. Manuel Ferreira
Fundação Planetário da Cidade do Rio de Janeiro
Condomínio do Edifício Morandi
Av. Padre Leonel Franca
Cms Pindaro de Carvalho Rodrigues
R. Félix Pacheco
R. Piratininga
Rua Frederico
Leonel França
Pontifícia Universidade Católica
Colégio Teresiano
Av. Padre Le
R. Embaixador Graça Aranha

Figure 3: Two M-Hubs, with one and three M-OBJs (SensorTags), displaying four (or five) current sensor readings.

As can be seen from Table 1 the Service discovery process in BLE is the heaviest operation. And since it will find all the services, characteristics and descriptors that the M-Object possesses, its delay will be proportional to the number of services, in our case, the 6 sensors. However, we found out that this Services Discovery time decreases very much for subsequent connections, as can be seen from Table 2. Hence, for subsequent connections, the total delay from the first contact to the data notifications takes less than 700 milliseconds.

Also, we also saw that the M-Hub almost immediately detects when a M-OBJ disconnects from it. Unfortunately, the precise latency of this detection - the callback of `onMObjectDisconnected()` in the `TechnologyListener` - could not be measured, but it is probably less than 500 milliseconds.

Further experiments were realized using two M-Hubs and four M-Objects (Sensor Tags). Here, we measured the amount of time that it takes for a server in the the SDDL Core to receive the sensor data from one M-Hub (MHUB2) that established a connection with all M-Objects (until services are discovered) since another M-hub (MHUB1) was disconnected from them. This time interval was measured at the server, from the moment it receives *M-OBJ DisconnectMessage* from MHUB1 until it receives a confirmation message that the M-OBJs services have been discovered. The same Motorola Moto X smartphone used in the first experiments was used as the MHUB1 and a Moto E with Android 4.4.4 KitKat was used as MHUB2. In these experiments, the scan interval was 0, so the scans

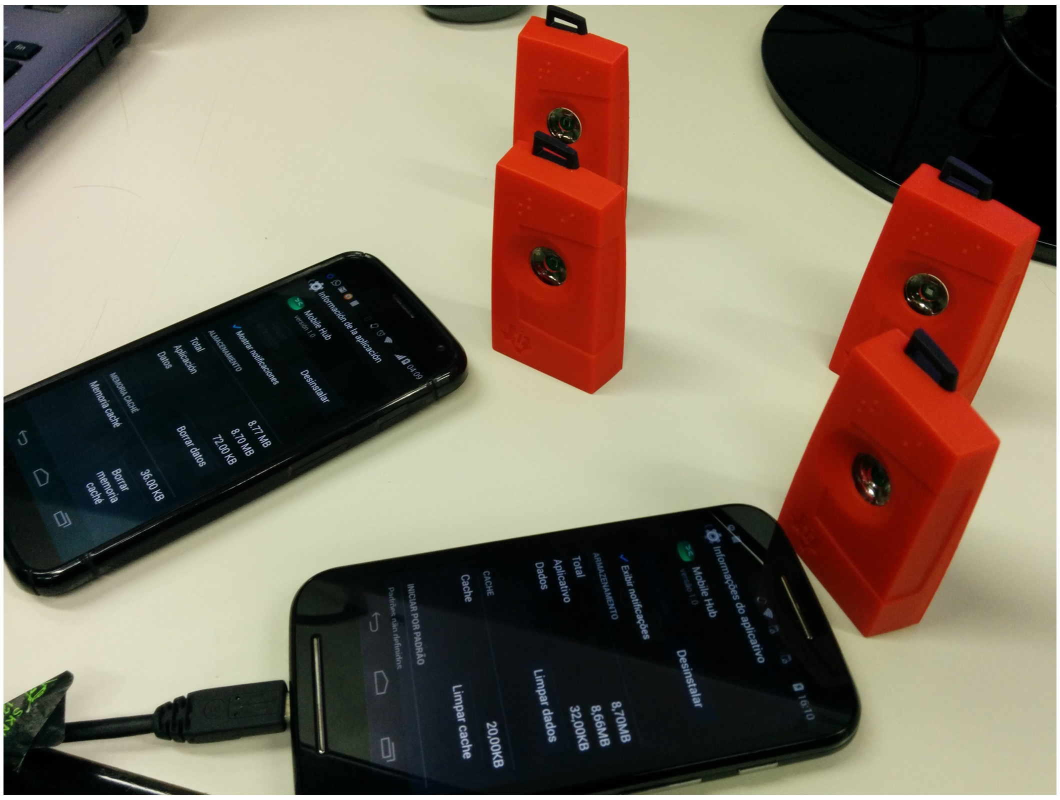
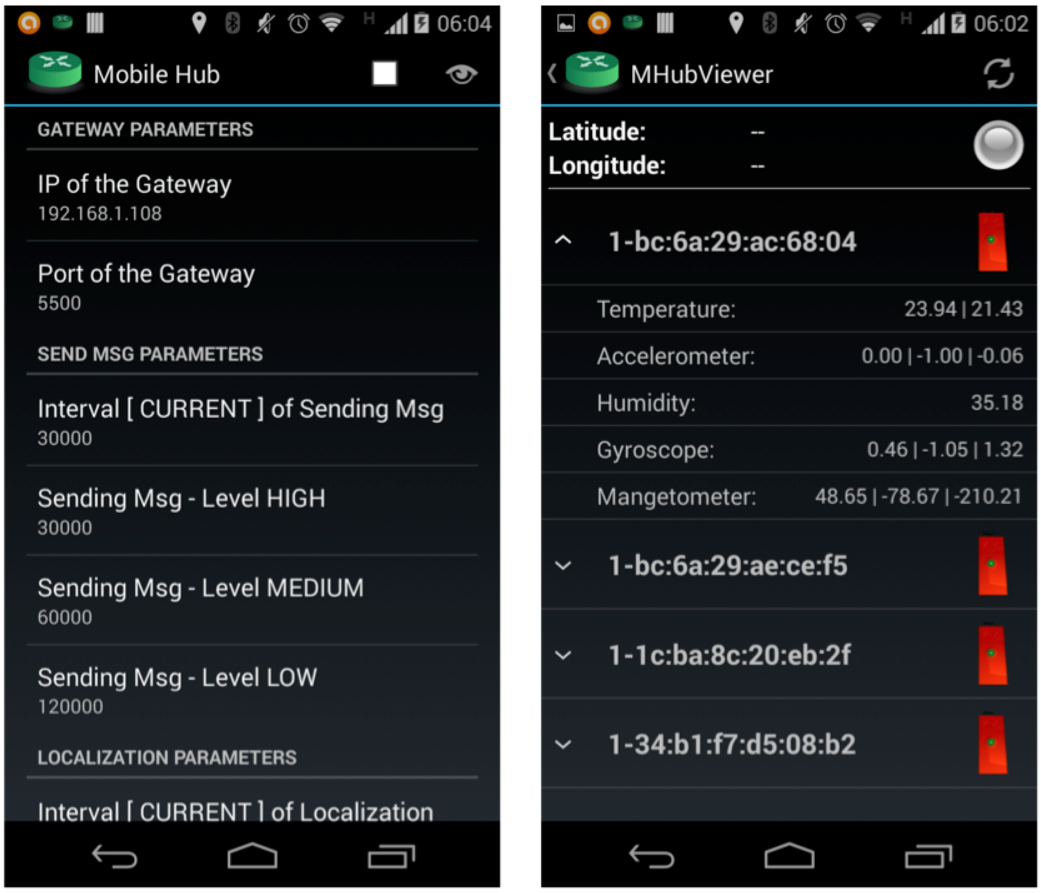


Figure 4: Two M-Hubs interacting with four M-OBJs during tests

started immediately after the stop of the previous scan. . Similar to the previous tables, the connections of the MHUB2, to all four M-OBJs, could be for a first one (Table 3) and for subsequent connections (Table 4). We ran each experiment 12 times and present the mean value and the standard deviation. It is also important to remark that the behavior of BLE in Android is synchronous, so the connections to each of the M-Objects is done sequentially.

The times in Tables 3 and 4 show that BLE enables a very fast *connection-plus-service discovery* latency of peripheral devices, specially if the M-Hub has already connected to them previously. So, if sensor data from M-OBJs has to be collected and transmitted to servers in almost real time by any M-Hub that happens to be its BLE wireless range, then it will suffice for an approaching (or moving by) M-Hub to stay at least 10 seconds nearby to the M-OBJ, and it will already start to receive and be able to relay the M-OBJ's sensor data to the server. Considering that BLE's communication range is approx. 20-30 meters, this means that if the relative speed between the M-Hub and the M-OBJ is slower than 3 m/s, then there is high probability that the M-Hub will transmit the sensor data to the server. And the chances are even higher, it there are more than one M-Hub in the vicinity of the M-OBJ.


Mobile Hub

GATEWAY PARAMETERS

IP of the Gateway
192.168.1.108

Port of the Gateway
5500

SEND MSG PARAMETERS

Interval [CURRENT] of Sending Msg
30000

Sending Msg - Level HIGH
30000

Sending Msg - Level MEDIUM
60000

Sending Msg - Level LOW
120000

LOCALIZATION PARAMETERS

Interval [CURRENT] of Localization

MHubViewer

Latitude: --
Longitude: --

1-bc:6a:29:ac:68:04

Temperature:	23.94 | 21.43
Accelerometer:	0.00 | -1.00 | -0.06
Humidity:	35.18
Gyroscope:	0.46 | -1.05 | 1.32
Mangetometer:	48.65 | -78.67 | -210.21

1-bc:6a:29:ae:ce:f5

1-1c:ba:8c:20:eb:2f

1-34:b1:f7:d5:08:b2

Figure 5: Screens of the MHub Viewer: M-Hub settings (left) and four discovered SensorTags with their current sensor readings (right)

7 Related Work

Most of the works, in both industry and academia, on service discovery protocols and connectivity for IoT in mobile environments are based on fixed communication infrastructures and centralized database servers. They utilize a client-server model, where mobile clients issue range queries to central query processing servers or brokers using protocols like MQTT over WiFi or ZigBee. In these systems, the stationary objects continuously transfer their data to a central server and clients can obtain information about near-by objects by issuing queries to the central server [7].

Some of the recent works address peer-to-peer and mesh network connectivity with service discovery in a mobile environment. There are industry protocols like Thread, from TheadGroup, AllJoin, from All Seen Aliance that are specialized on support of discovery and interoperability of things across devices and platforms in a specific environment (e.g. home, office), and academia protocols designed to be mobile and energy efficient by grouping devices in a neighborhood such that devices in a group will take turns to announce the

Table 1: First Connection to M-Object (BLE)

Run	CoT (s)	SDT (s)	ENT (s)
1	0.314	9.159	0.430
2	0.393	8.960	0.498
3	0.201	9.157	0.311
4	0.102	9.132	0.214
5	0.045	9.411	0.158
6	0.605	8.815	0.723
7	0.128	9.232	0.237
8	0.197	9.094	0.306
9	0.094	9.188	0.206
10	0.143	9.164	0.258
11	0.111	9.172	0.228
12	0.210	9.100	0.331
Mean	0.21192	9.132	0.325
St.Dev	0.1578	0.14381	0.15839

existence of other devices in a group [6]. There are also technologies like Bluetooth Service Discovery Protocol (BSDP) [1], Universal Plug and Play (UPnP) [8, 4] and Secure Service Discovery Service (SSDS) [2] as a specification which enables close devises to communicate with each other at low cost and low power consumption, defines interactions among smart object, people, and environments and also supports authentication, privacy, and integrity.

Many efforts also have been based on the CoAP protocol. CoAP natively provides a mechanism for service discovery and location [10]. Each CoAP server must expose an interface to which the generic node can send requests for discovering available resources. The CoAP server will reply with the list of resources and, for each resource, with an attribute that specifies the format of the data associated to that resource. CoAP, however, does not specify how a node joining the network for the first time or how it must behave in order to announce itself to the resource directory node.

Although there are many protocols for Internet of Thing, many of them didn't embrace mobile nodes, do not consider mobile or movable smart objects, or do not scale. We are unaware of a comprehensive approach focused on the Internet of *Mobile Things*, in which the connectable things can be moved or can move independently, and yet remain remotely accessible and controllable from anywhere in the Internet with handover. Nor have we seen any work showing the design, concrete prototype implementation and experiments of a Mobile Hub. with a concrete WPAN technology. The only work that presents a similar idea of using the smartphone as an IoT Gateway is [5], but their software architecture for the smartphone is rather high-level, uses traditional protocols (e.g., TCP, UDP), and does not consider any concrete short-range, low-power WPAN or WLAN technology.

8 Discussion

We believe that for future IoT applications, users and developers should not need to know the URI of individual smart things. The mere scale of the problems, involving

Table 2: Subsequent Connections to M-Object (BLE)

Run	CoT(s)	SDT (s)	ENT (s)
1	0.077	0.158	0.193
2	0.395	0.161	0.496
3	0.164	0.165	0.268
4	0.102	0.126	0.210
5	0.256	0.165	0.360
6	0.316	0.179	0.423
7	0.161	0.134	0.359
8	0.153	0.159	0.259
9	0.315	0.153	0.423
10	0.316	0.133	0.422
11	0.231	0.170	0.335
12	0.304	0.159	0.407
Mean	0.2325	0.15517	0.34625
St.Dev	0.10007	0.01609	0.0953

thousands of things and millions of sensors and actuators will make such system impossible to manageable at the level of things. Instead, what will matter for the applications are just the data, their context and the semantically rich information that can be derived from this data. For example, instead of subscribing for a data stream from a specific sensor, the IoT applications will probably issue more abstract queries or subscriptions like: "mean pressure value at a given place", or "is there some discrepancy of the temperature readings within a building", or else "show me the air pollution indices along my biking route".

Hence, we think that future IoT applications will require the following two things:

1. a means of enriching the collected data on its path from the source/s to the consumers (i.e. the application or users);

2. ways of discovering, selecting, subscribing or querying these data based on high-level filtering and correlation expressions in terms of the data's semantics and its context information.

In this sense, the Mobile Hub concept supports item (1), since it not only adds Internet connectivity to the otherwise isolated smart things, but is also capable of executing basic aggregation, summarization and transformation functions on the sensor data, as well as enrich this data with contextual information, such as the approximate geographic position of the sensor; if it is moving and how fast; which is the current sound level (i.e., activity) of the environment; whether there is some other sensors in the vicinity that can provide complementary or confirmatory information, etc.

Moreover, the M-Hub also enables tangible anywhere-accessible user interfaces for managing (configuration, state monitoring, operation control, etc.) of sensors and control units embedded in smart things (and Mobile Objects in particular), that lack an own user interface.

Table 3: Handover for First Connections to all M-Objects (BLE), in seconds

Run	1 M-OBJ	2 M-OBJs	3 M-OBJs	4 M-OBJs
1	10.307	20.045	29.693	39.503
2	11.034	20.804	30.437	40.283
3	11.593	21.079	30.909	40.666
4	9.286	18.856	28.733	38.605
5	11.679	21.459	31.312	41.381
6	9.357	19.309	29.288	38.940
7	9.243	18.957	28.784	38.490
8	9.400	19.047	28.901	38.888
9	9.401	19.038	28.974	39.004
10	11.570	21.120	31.082	40.805
11	11.043	20.562	30.487	40.122
12	9.980	19.705	31.607	41.182
Mean	10.32442	19.99842	30.01725	39.82242
St.Dev	1.00115	0.96796	1.07327	1.04175

9 Conclusion

We have presented the concept, design and a prototype implementation of the Mobile Hub, as the main enabler for the Internet of Mobile Things (IoMT), where any smart thing/object with some short-range and low-power wireless interface (WPAN technology) is movable and nevertheless can be connected to the Internet. We have also presented some hypothetic applications for IoMT to illustrate the potential benefit of this new IoT paradigm. The M-Hub concept is independent of the mobile platform and has been designed to be used with several WPAN technologies, since its constituent S2PA Service defines a generic Technology Interface that has to be implemented for each WPAN technology.

However, our first prototype has been implemented for Android only, uses our mobile communication middleware SDDL, and has used Bluetooth Low Energy (BLE) as the showcase WPAN. Actually, because of the high efficiency of this WPAN technology, this turned out to be an excellent choice. Yet another advantage, is that BLE is being widely adopted for smart objects, and is being supported by most smartphone makers. For IoMT, where fast connection with objects and discovery of their (sensor/actuation) services will be paramount because of the fast variability of the relative distance between Mobile Hubs and its nearby mobile objects, Bluetooth LE showed excellent results for discovery, reconnection and handover of BLE-enabled M-OBJs. For example, if a M-Hub and a M-OBJ have already discovered each other before, the subsequent reconnection takes 0.9 seconds, and can be done for four M-OBJs in less than 5 seconds. Moreover, these latencies are strongly affected by Android's KitKat current support for Bluetooth LE, which implements just sequential connections and service discovery of M-OBJs. We have made all our experiments with four BLE-enabled SensorTags, and have shown how their sensor data can be instantly received by the M-Hub and then relayed to any server/monitor

Table 4: Handover for Subsequent Connections to all M-Objects (BLE), in seconds

Run	1 M-OBJ	2 M-OBJs	3 M-OBJs	4 M-OBJs
1	0.379	1.640	2.104	3.350
2	0.212	2.237	3.535	4.252
3	0.372	0.915	1.688	2.311
4	2.936	3.852	4.856	5.468
5	0.033	1.103	1.956	2.274
6	0.426	1.010	1.639	2.454
7	1.992	3.016	3.629	4.448
8	0.393	1.507	2.074	3.018
9	0.535	4.444	5.217	6.021
10	2.192	2.861	3.775	4.388
11	1.361	2.176	3.000	3.782
12	0.271	0.920	1.626	2.235
Mean	0.92517	2.14008	2.94992	3.66675
St.Dev.	0.95336	1.18817	1.25685	1.28251

through the M-Hub's Internet connection. We have also testes other Android apps[8] with some similar capability to connect and receive sensor data from BLE objects. However, none of them supports such fast transfer of data to the internet, and also is restricted to connecting to a single BLE at a time.

In spite of the encouraging preliminary results, we are aware that our current M-Hub prototype is only ”scratching the surface of IoMT”, and much interesting research, software development and applications can be derived from this work. In particular, our future work includes: investigate the problems and possible approaches inter- M-Hub handover protocols aiming to reduce the latency of the first connections with M-OBJs, including an local *Data (Stream) Processing Service* into the M-Hub, so that it is capable of processing the sensor data received from the M-OBJs, so as to correlate it and infer higher-level information from the sensed data stream before sending this information via its Internet connection. Moreover, we also plan to study means of sending commands to M-OBJs with actuators, and thus support any Internet-wide remote control of smart things, such as home appliances. In particular, we also want to study if the local local Data Stream Processing Service can be used for local feedback-control-loops on sets of M-OBjs with sensors and actuators.

References

[1] BluetoothSIG. Specification of the bluetooth system – core. Available online.

[2] S. Czerwinski, B. Y. Zhao, T. Hodes, A. Joseph, , and R. Katz. An architecture for a secure service discovery service. In *Fifth Annual International Conference on Mobile Computing and Networks (MobiCom '99)*, 1999.

[8]SenseView - http://senseview.mobi

[3] F. DaCosta. *Rethinking the Internet of Things: a scalable approach to connecting everything*. ApressOpen, New York, NY, 2013.

[4] U. Forum". Universal plug and play device architecture 1.0, 2013.

[5] R. Golchay, F. L. Mouël, and S. Frénot. Towards bridging iot and cloud services: Proposing smartphones as mobile and autonomic service gateways. *arXiv preprint arXiv*, 1107.4786., 2011.

[6] P. Huang, E. Qi, M. Park, and A. Stephens. Energy efficient and scalable device-to-device discovery protocol with fast discovery. In *IEEE International Conference on Sensing, Communications and Networking (SECON 2013)*, pages 1–9, 2013.

[7] U. Hunkeler, H. Truong, and A. Stanford-Clark. Mqtt-s - a publish/subscribe protocol for wireless sensor networks. In *3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE)*, pages 791–798, 2008.

[8] B. A. Miller, T. Nixon, C. Tai, and M. D. Wood. Home networking with universal plug and play. *IEEE Communications Magazine*, pages 104–109, 2001.

[9] G. Pardo-Castellote. Omg data-distribution service: Architectural overview. In *Proceedings of the 2003 IEEE Conference on Military Communications - Volume I*, MILCOM'03, pages 242–247, Washington, DC, USA, 2003. IEEE Computer Society.

[10] Z. Shelby, K. Hartke, , and C. Bormann. Constrained application protocol (coap), 2013.

[11] L. Silva, R. Vasconcelos, L. Alves, R. Andre, and M. Endler. A DDS-based middleware for scalable tracking, communication and collaboration of mobile nodes. *Journal of Internet Services and Applications*, 4(1):1–15, 2013.

[12] L. D. Silva, M. Endler, and M. Roriz. Mr-udp: yet another reliable user datagram protocol, now for mobile nodes. Monografias em Ciencia da Computacao 06/2013, Departamento de Informatica, PUC-Rio, May 2013.

[13] I. Vasconcelos, R. Vasconcelos, and M. Talavera, L. Endler. Towards a model and middleware for the internet of mobile things. In *Wireless Days (to appear)*, 2014.

[14] R. Vasconcelos, M. Endler, B. Gomes, and F. Silva. Autonomous load balancing of data stream processing and mobile communications in scalable data distribution systems. *International Journal On Advances in Intelligent Systems*, 6(3-4):300–317, 2013.

[15] R. O. Vasconcelos, L. Silva, and M. Endler. Towards efficient group management and communication for large-scale mobile applications. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2014 IEEE International Conference on*, pages 551–556, March 2014.