Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 63 (2015) 112 – 119

Procedia
Computer Science

The 6th International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN 2015)

A Framework for Handling Heterogeneous M2M Traffic

Asma Elmangoush[a]*, Andreea Ancuta Corici[b], Ronald Steinke[b], Marius Corici[b], Thomas Magedanz[a]

[a]*Next Generation Networks, Technical University Berlin, Berlin, Germany*
[b]*Next Generation Network Infrastructures, Fraunhofer FOKUS Institute, Berlin, Germany*

Abstract

Sensors, actuators and devices that compose the Internet of Things (IoT) world are becoming more diverse every day in terms of capabilities and amount of generated traffic. Current Machine-to-Machine (M2M) communication standardization efforts try to formalize the interfaces between M2M nodes based on the perspective of exchanging uniform small data size with low sampling rate only. However, many devices will require support for more heterogeneous traffic patterns, with different network capacity. This paper introduces a communication concept for supporting gracefully a heterogeneous set of devices. This paper analyses the effect of traffic size in M2M transactions and propose a concept to adapt gracefully to support heterogeneous traffic patterns in M2M systems. To prove its feasibility, the concept is exemplified on top of oneM2M architecture and implemented as part of the Fraunhofer FOKUS OpenMTC toolkit. Additionally, the concept was applied to a deployment in an E-Health pilot and practical measurements during functional evaluation are reported.

Keywords: Framework; M2M traffic; E-Health; Communication Management; Telemonitoring

1. Introduction

The connected world is extending continuously to include physical objects besides computers and smartphones in a global Internet of Things (IoT). Machin-to-Machine (M2M) communication shall enable every physical and virtual object to integrate seamlessly into a large-scale infrastructure without human intervention. This applies and reinforces

* Corresponding author. Tel.: +49-30-34637106 ; fax: +49-30-34638000.
E-mail address: asma.a.elmangoush@campus.tu-berlin.de

doi:10.1016/j.procs.2015.08.319

a convergence of heterogeneous technology families in various layers (application, transportation and physical) to accommodate the technical communication and service requirements.

The M2M services communication requirements was believed to refer to M2M traffic as small chunks of data in a low transmission rate, mostly with predictable communication times and duration[1,2]. However, M2M services have to integrate gracefully all types of devices measuring or actuating in the physical world. This includes devices requiring real time transmission for time-constraint applications, as well as delay tolerant transmission for batch processing applications. For example, in remote surgery-theater monitoring the sampling rate and complexity of devices on both end systems are very high and require synchronization of data sources as well.

Furthermore, many M2M components integrated in a Smart City framework will execute multiple applications from different domains in order to minimize the number of deployed devices and gateways. This results in using the same component for multiple application domains, and therefore the adaptability of these components based on the applications requirements is essential. The challenge is to build a connectivity aware M2M node that perform fairly and secure communication interweaving. Different transport protocols for M2M and application's communication requirements has been discussed in our previous work[3]. This paper presents a novel concept for efficiently handling the heterogeneous M2M communication requirements into a comprehensive system. First, the data transmission patterns are classified into different classes using the criteria of data size, rate of data acquisition and urgency of reception. Then, a generic solution is proposed to optimize the M2M connectivity and data delivery mechanism by introducing transmission session control.

The rest of the paper is structured as follows: Section 2 reviews related work in M2M; in Section 3 the analysis of the transmission's types and the proposed solution are described. Section 4 presents the system architecture and undergone deployment. Section 5 presents the proof of concepts evaluation results. Finally, we conclude the paper in Section 6.

2. Background and Related Work

Many standard organizations have promoted several activities in the M2M communication domain. The M2M standardization by the European Telecommunications Standards Institute (ETSI)[2] was based on service requirement for M2M in different use cases, such as Smart Metering and e-Health. Recently, ETSI M2M work is transferred to oneM2M consortium[4] aiming to consolidate the standardization work in M2M communications and develop technical specifications addressing the needs for a common M2M service layer. ETSI M2M and oneM2M refer to the REST-full application-level transport protocols: HTTP[5] and the Constrained Application Protocol (CoAP)[6]. While HTTP fits the requirements for web transfers, it requires a TCP session established consuming valuable energy and computation capacity. CoAP is proposed by the CORE (Constrained RESTful Environments) IETF working group to accommodate M2M devices with constrained resources in the integration with existing web services. It uses UDP for transportation and has a low overhead. The disadvantage of CoAP is that the requests might not be processed without retransmission.

The 3GPP refers to M2M communication as Machine Type Communication (MTC) and started standardization activities in September 2008 as part of 3GPP Rel-10 specifications. The service requirements working group (3GPP SA WG1) had specified a number of use cases and scenarios, and derived a set of service requirements accordingly[1]. In 3GPP Rel-11, some of the proposed MTC features were finalized, such as addressing and device triggering. The inclusion of M2M communication in 5G involves satisfying the fundamentally different requirements associated with multiple traffic classes, and massive number of devices in various circumstances. New methods at both the component and architectural levels are required to address these requirements[7].

Handling access management of M2M communication over wireless channels takes part at the MAC layer, some work is taking the direction of proposing hybrid MAC layer to enable the interworking. E.g., authors of[8] proposed a scalable hybrid MAC protocol for heterogeneous M2M networks, with the aim to combine the benefit of both contention-based and reservation-based protocols. Other work area tries to solve the problem in the routing level[9]. In paper[10], a group communication mechanism is proposed that manages data in up and down link, sleep schedule, energy of the network and device management. The proposed component aims to allow multiple M2M applications to use efficiently the network resources according to the applications' needs, the network overload level and the device resources.

3. Communication Requirements

In this section, we analyze the possible traffic demands for M2M applications in the Health care domain and approximately classify data delivery requests using message size and data rate in four categories from C1 to C4 as illustrated in Fig. 1. Numerous M2M services are considered to generate small size of messages (less than 1 Kbyte) in variant samplings rates, which are presented in C1 to C3 categories. However, a number of cases will generate much bigger messages (in Mbytes) but with relative low frequency, these are included in C4 category.

During a remote monitoring session of multiple vital signs, the data size of one property measurement is in the range of a few bytes[11]. In the case of an emergency where actuators can be used to administrate medicine, the complexity of the system and the precision required for rapid actuate can trigger a high request rate for a careful monitoring in critical timing. This case is covered by category C1 that include small size of data generated in a high frequency rate. Generally, data generated by sensors used for monitoring services are included here. The C2 category reflects the traffic generated in M2M services that produce small message size with slightly low frequency rate, such as once per second. This is the most expected kind of traffic in M2M systems. In many cases, the sensors might not have continuous Internet connectivity, due to their mobility feature or their limited capabilities. In such cases, the M2M applications have to adapt its performance. One option could be storing data locally and uploading to main server when the connectivity is re-established, instead of sending data instantly. Other option is to compute some statistics on the data, and send later the results. The statistical data volume depends on the level of granularity to be preserved on the statistics and the number and type of sensor data to be considered in the statistics. All this data will have to be transmitted once the connection is back again, and their importance might force their immediate transmission when an action should be taken based on the carried information. Even in case of reliable connectivity, the M2M application might follow a more energy efficient behavior by connecting from time to time and transmit statistics only. For such a category (C4) with big data size and low rate, an example would be e-Health statistics covering one week of data aggregation about activities timing, amount of exercise and consumed food with estimated calories.

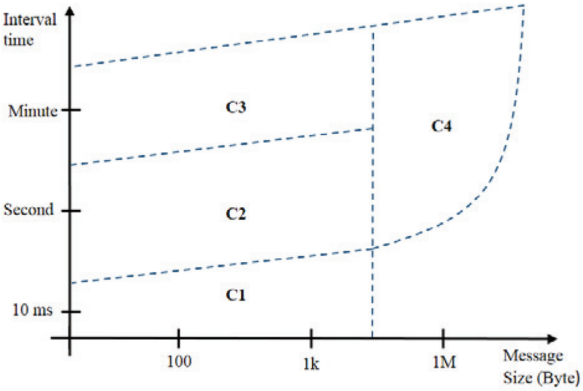

Fig. 1. Classification of M2M data delivery according to data size and sampling interval

4. The Multiple Data Flow Management (MDFM) Concept

The workflow performed by a typical M2M system starts with sensors acquiring data from the surrounding environment (e.g. temperature, humidity etc.). Then, processing the data in a suitable format to be delivered to a decision-making application via a M2M middleware platform. The application shall perform different level of data analysis and send commands that trigger actuation tasks (e.g., alerts/information, or commands to actuators, or relevant events) based on the output of the analysis process.

As discussed in previous subsection, the size of the data exchanged between M2M nodes varies according to the sensor type and service's requirements. In addition, the final size of data unit depends on data encoding and encryption technologies used by the selected communication path. In our previous work[3], we discussed and evaluated the effect of message size in the performance of M2M applications. Although CoAP has the advantage of a small overhead of only 4 bytes, the encoding/decoding time was slightly affected with the size of the payload. A block-wise transfer mechanism was proposed in the IETF draft of CoAP[12] that enables the negotiation of the block size. This procedure is repeated every time a new data measurement takes place. This leads to an inefficient data transmission as the application is not aware of the optimum data size to be used for transmission. Although the transportation engine defined by oneM2M as Communication Management Delivery Handling (CMDH) function can use policies to select communication path, it is not aware of the data sampling rate or the priority of the data to be sent. Making the CMDH aware of all the possible data formats would make the transmission complexity impossible to manage.

We propose a new class of Resources named BufferResource (see Fig. 2) designed to store data with different associated transmission criteria information: transport priority class, transmission delay tolerance or other transmission criteria like synchronization between chunks (useful in real time flow synchronization). In order to enable long-term information exchange between the communication modules of the M2M Gateway and M2M server endpoints, the application starts a transmission session in the communication module. The application can indicate at session establishment the initial transmission criteria of the data to be sent. During session establishment, the application will refer to the BufferResource and the communication module will monitor it, being aware of the whole available data to be sent. The communication module will organize the data into multiple data flows, one flow for each priority class. For example, the data that detect an alarm or emergence case are usually in small size and should have higher priority than other type of traffic.

The flows will only become active if instructed from the application by indicating transmission criteria characterizing the type of data to be sent. For example, in case the user or the back end application would request sending more details through methods out of scope of this paper, the front-end application will indicate to the CMDH to use another minimum priority class of the data in order to enable lower priority flows. If by enabling new flows that contain considerable amount of data would render the current communication path congested or too costly to accommodate the new flows, then another communication path might be used. This ensures that data with higher priority can reach the target in time and the lower priority data containing details would still reach the target with some delay.

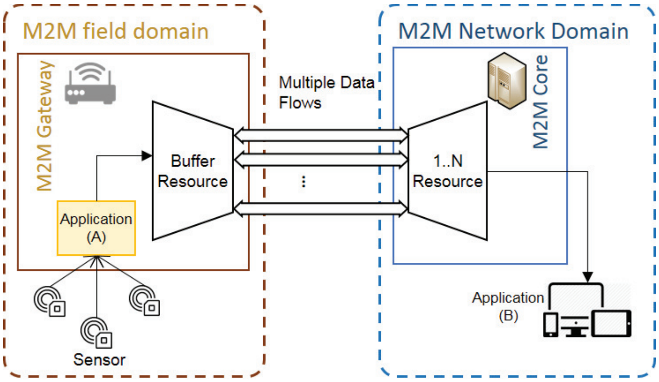

Fig. 2. Distribution of BufferResource in multiple Data Flows

5. System Architecture and Implementation

The OpenMTC platform[13] is a prototype implementation of oneM2M standard specification, designed to act as a horizontal convergence M2M layer supporting multiple vertical domains[14]. The architecture of the OpenMTC is compatible with the high-level functional architecture specified by oneM2M[15]. The field domain includes Middle Nodes (MN) and Application Service Nodes (ASNs), which act as M2M gateways to handle the interaction with multiple connected sensors. Each node contains a Common Service Entity (CSE) that instantiate a set of functions for the M2M environment. In the infrastructure domain, the M2M core server contains one CSE that communicate via the Mcc interface with the MNs and ASNs at the field domain. Application Entities (AEs) represents the logic of the end-to-end solution, and responsible to process the data collected from the sensors to store it in the Data Management & Repository (DMR) function. It is the responsibility of the Communication Management and Delivery Handling (CMDH) function to select the communication path to destination. Other CSE functions are deployed in the nodes, however due to the short of space we will not describe them here. Interested readers could refer to[15] for further information.

The concept of multiple data flow management was implemented on the Fraunhofer FOKUS OpenMTC toolkit, designed for fast prototyping innovation in M2M domain. The framework is implemented in Python 2.7. The modules are designed to run on an event-based platform based on Gevent[16], which is an event based library providing asynchronous I/O API that can scale its number of execution units according to the processing load. In Fig 3, the inner-architected of the OpenMTC is illustrated. The main functionality defined by the oneM2M CSEs are developed in the core with a set of application enablement APIs to support the devolvement of application by 3rd party. Optional features are included as plugins that share an inner-API based on events to communicate with the OpenMTC Core. An internal subscribe/notify mechanism enables plugins to be notified about updated statues. The Adaptable M2M Transport (AdM2M) provides different transport protocol stacks, such as HTTP and CoAP as plugins to support the communication with M2M devices using the proper stack to each use case. This is controlled by the Connectivity Manager operations to enable and disable interfaces depending on the operating system (Android or Linux).

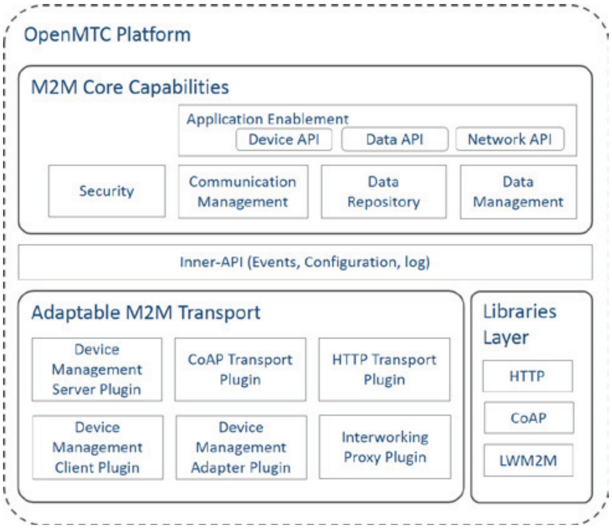

Fig. 3. The OpenMTC platform Architecture.

To implement the proposed concept, described in previous section, the CMDH shall force policies according to the request parameters set by applications (i.e. delay tolerance and priority). As an extension to the oneM2M architecture, we introduce two new extend functionality to the CMDH: the Data Transmission Session Manager (DTSM) and the Connectivity Manager (CM). The DTSM handles the session establishment, termination and modification of the data transmission requested by the application as presented by the attributes of BufferResource, which is stored in the DMR. This is achieved by distributing the data into multiple flows. The CM is responsible to handle the communication channels over the bearer networks.

6. Proof of Concept Evaluation

Local network tests were performed to prove the feasibility of the implementation using CoAP as transport protocol. Two machines were used to send and receive requests, both running with Intel Core 2 Quad CPU (2.4 GHz) and 4GB of RAM. Requests were sent over 100Mbit/s Ethernet with a default MTU set to 1500 bytes. An SQLite3 database was used for the initial measurement. In the current work we have made proof of concept tests to observe the behavior by excluding radio collision, tests with heterogeneous access networks are envisioned in future work.

6.1. Evaluating the Effect of BlockSize

The first set of functional measurements were conducted to prove the implementations of block-wise transmission and observe the effect of the used block size on data flows with fixed interval times and payload size. The average response time of confirmable CoAP POST requests, used to push data towards the M2M server node, were measured in each case. In each test, a flow of post requests containing a payload size, ranging from 20 bytes to 10Kbytes is sent to the M2M server. The interval time of the requests were 0,5sec (2 req/s), 0.01s (100req/s) and 0,004s (250req/s). Fig. 4 shows the average response times using the block size of 64 bytes (SZX=4) and 1024 byte (SZX=6) with the Post requests in each case. The plot illustrates the impact of the increased payload size clearly.

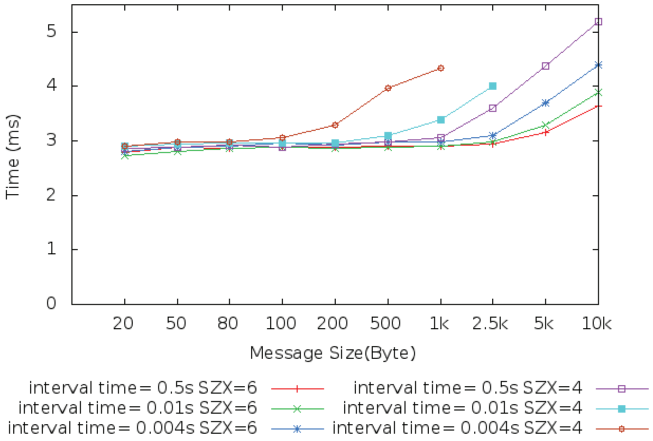

Fig. 4. Avarage response time for different request rates, with MTU of 1500 and block size of 64 bytes and 1024 bytes for each rete

We note that the response time increases for requests rate of 100req/s rate or higher with payload bigger than 1Kbytes. These results reinstate that the block size has to be adapted to the available MTU and payload size. In the case of access networks with limited MTU, transferring a large amount of data with a rate greater more than 100 request/s could be critical. Another aspect to consider in these tests was the loss rate of requests. As can be seen in Fig 5, the rate of lost or timed out requests is increasing linearly with the request rate a loss rate of 8.5 % for a flow of 1500 requests per second. In oneM2M specification of the protocol binding for CoAP, the blockwise transfer is recommended to be used to transfer large amount of data, e.g. for firmware update. Based on these results, a recommendation of using 1Kbyte as blocksize for update downloading, or similar transaction scenarios, could be concluded in order to reduce the transmission delay and avoid retransmitting requests.

Finally, the implementation was tested to compare the performance of splitting a data stream into multiple flows. We analyze that by breaking down a data stream into several flows in order to compare the performance in both cases and send the data to destination in proper time. In case of splitting a traffic flow of 250 req/s into five parallel flows of 50 req/s each, with the same payload size of 100Byte. The average delay for the single data flow was 7ms, while in the case of five parallel flows the average delay decreased to 4.3ms. Additionally, the percentage of retransmitting lost packets was decreased from 1% to 0.3%. While only minor changes in the consumption of CPU resources was remarked, the bandwidth usage have increased by 80%. This proves the concept of splitting data into multiple flows in order to send an important amount of data with high generation frequency. The data could be sent with similar rate and smaller data size without overloading the server.

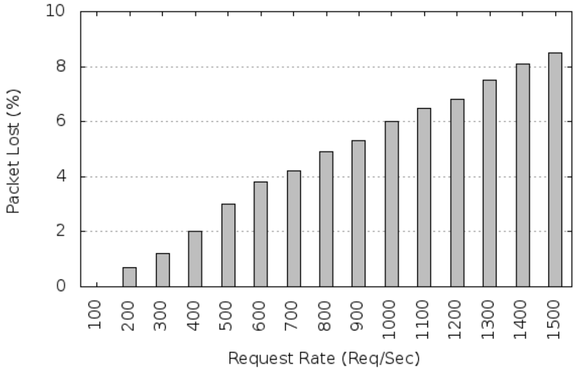

Fig. 5. Rate of lost requests for one flow

6.2. EHealth Scenario

In this subsection, we will present the results of testing the concept of using multiple flows of data streams in an EHealth scenario, where the user uses an attached sensor to monitor his/her heart ECG signals. The data is stored in the user-owned gateway that implement some data analysis to detect an emergence case. In addition, a set of accumulated data shall be send to the central server once per day for further analysis, and it is possible that the gateway receive a software update, when available, from the service provider. This backup or update traffic has less priority than the vital signs, but will produce huge amount of traffic with larger payload messages.

The measurements in Fig 6 plots the rolling average for the response time of the Post requests used to push the data from the sensor adapter to the gateway. The ECG signals are generated in a data rate of 250 requests per second and a fixed payload size of 20Byte. The measurements show three scenarios, the first one is when the data is push without the existence of any background traffic in/out the gateway. In the second scenario, an additional data flow is taking place that present firmware update downloading to the gateway on a low rate (1 Post request/s, 5Kbyte payload). The third scenario is similar but the DTSM is splitting the update flow into two flows. From the plot, we observe the performance improvement of the proposed concept.

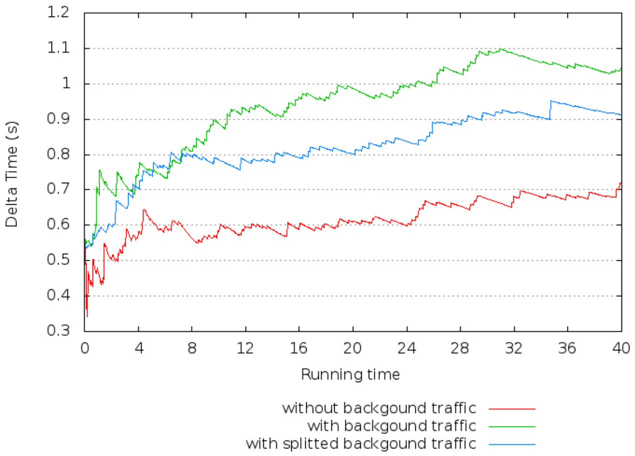

Fig. 6. Rolling average of the transport delay for EHealth case.

As the server is receiving a cumulated load, the delay was higher with the existing of the update flows. We can even observe that if the flow has as increased request rate, it has greater average response delay. The possible cause could be because the number of open threads on the server is slightly over provisioned to overcome the load of the requests. The DTSM could schedule data with high priority on flows with increased request rate and data with lower priority with lower request rate without a critical delay increase, so that all data can be sent in a fairly manner without being stalled. The important thing is to take into account the current delay, in order that the request rate can be adapted to the registered delay.

7. Conclusion and Future Work

The characteristics of the M2M traffic are different from the existing Human-to-Human and Human-to-Machine traffic. Due to the heterogeneity of connecting objects in terms of capability and generated traffic, the M2M middleware needs to adapt its handling to the M2M traffic based in the application requirements and traffic pattern. In this paper, we present a conceptual classification of M2M traffic profiles in order to gain a better understating of the communication requirements of such services. Additionally, we propose a novel concept aiming to handle the heterogeneous traffic from embedded devices/sensors in the IoT. As further steps we plan to investigate the correlation of the average used CPU of the M2M server with the statistics on the request rate and data size transmission carried during the tests. Another investigated concept is to enable Communication Management and Delivery Handling policies usage to route a message via multiple hops. Some of the hops can be used to secure the transmission and other for enabling load balancing of multiple back-end application instances or brokerage between multiple application domains. The concept was implemented as part of the Fraunhofer FOKUS OpenMTC toolkit. Additionally, the concept was applied to a deployment in an E-Health pilot and practical measurements during functional evaluation are reported.

References

1. GPP-TS22.368. Service requirements for Machine-Type Communications (MTC) Rel-12. 3GPP; 2013.
2. ETSI TS 102 689 V1.1.2. Machine-to-Machine communications (M2M); M2M service requirements. 2011;
3. Elmangoush A, Steinke R, Magedanz T, Corici AA, Bourreau A, Al-Hezmi A. Application-derived communication protocol selection in M2M platforms for smart cities. 2015 18th International Conference on Intelligence in Next Generation Networks [Internet]. Paris, France: IEEE; 2015. p. 76–82. Available from: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7073810
4. oneM2M. oneM2M Requirements Technical Specification. 2013.
5. IETF RFC 2616: Hypertext Transfer Protocol [Internet]. 1999. Available from: https://datatracker.ietf.org/doc/rfc2616/
6. Shelby Z, Hartke K, Bormann C. RFC 7252: The Constrained Application Protocol (CoAP). 2014.
7. Boccardi F, Heath RW, Lozano A, Marzetta TL, Popovski P. Five disruptive technology directions for 5G. IEEE Commun Mag [Internet]. 2014 Feb [cited 2014 Apr 30];52(2):74–80. Available from: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6736746
8. Liu Y, Yuen C, Cao X, Hassan NU, Chen J. Design of a Scalable Hybrid MAC Protocol for Heterogeneous M2M Networks. IEEE Internet Things J [Internet]. 2014 Feb [cited 2015 Feb 3];1(1):99–111. Available from: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6762845
9. Wang G, Leng J, Bai L. A game-theoretic analysis of multiple protocol data flows in hierarchical M2M communication networks. Int J Distrib Sens Networks [Internet]. 2013;2013. Available from: http://www.hindawi.com/journals/ijdsn/2013/584707/
10. Jiann-Liang Chen, Han-Chuan Hsieh, Larosa YT. Congestion control optimization of M2M in LTE networks. Advanced Communication Technology (ICACT), 2013 15th International Conference on. 2013. p. 823–7.
11. Zephyr BioHarness [Internet]. Available from: http://zephyranywhere.com/products/bioharness-3/
12. Shelby Z, Bormann C. Block-wise transfers in CoAP [Internet]. 2015 [cited 2015 Apr 27]. Available from: https://tools.ietf.org/html/draft-ietf-core-block-17
13. OpenMTC platform [Internet]. Available from: http://www.open-mtc.org/index.html
14. Elmangoush A, Al-Hezmi A, Magedanz T. The Development of M2M Standards for Ubiquitous Sensing Service Layer. 2014 IEEE Globecom Workshops (GC Wkshps) [Internet]. IEEE; 2014 [cited 2015 Mar 26]. p. 624–9. Available from: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7063502
15. oneM2M-TS-0001. OneM2M Functional Architecture. 2015.
16. Gevent, Python network library [Internet]. Available from: http://www.gevent.org