A Service Based Architecture for Multidisciplinary IoT Experiments with Crowdsourced Resources

Panagiotis Alexandrou[1,2], Constantinos Marios Angelopoulos[3,7], Orestis Evangelatos[3], João Fernandes[5], Gabriel Filios[1,2], Marios Karagiannis[3], Nikolaos Loumis[6], Sotiris Nikoletseas[1,2], Aleksandra Rankov[4], Theofanis P. Raptis[1,2], José Rolim[3], and Alexandros Souroulagkas[1,2]

[1] Computer Engineering and Informatics Department, University of Patras, Greece
[2] Computer Technology Institute and Press "Diophantus", Patras, Greece
[3] University of Geneva, Switzerland
[4] DunavNET, Novi Sad, Serbia
[5] Alexandra Institute, Aarhus, Denmark
[6] University of Surrey, Guildford, UK
[7] Bournemouth University, UK

aleksandro@ceid.upatras.gr

Abstract. Research on emerging networking paradigms, such as Mobile Crowdsensing Systems, requires new types of experiments to be conducted and an increasing spectrum of devices to be supported by experimenting facilities. In this work, we present a service based architecture for IoT testbeds which (a) exposes the operations of a testbed as services by following the Testbed as a Service (TBaaS) paradigm; (b) enables diverse facilities to be federated in a scalable and standardized way and (c) enables the seamless integration of crowdsourced resources (e.g. smartphones and wearables) and their abstraction as regular IoT resources. The architecture enables an experimenter to access a diverse set of resources and orchestrate experiments via a common interface by hiding the underlying heterogeneity and complexity. This way, the field of IoT experimentation with real resources is further promoted and broadened to also address researchers from other fields and disciplines.

Keywords: Internet of Things, Testbeds, Architectures, Platforms, Crowd

1 Introduction

Experimental facilities also known as testbeds, provide the controlled environment needed for implementing and testing novel technologies and architectures. Among others things, they follow agile architectures that are easy to re-configure in the context of experiments and provide additional services and tools for collecting meta-information on the experiment execution (e.g. monitoring several performance metrics or providing execution logs for post-experiment processing).

Also, their controlled environment constitutes a base reference that helps in evaluating and comparing different architectures and protocols. However, despite the services and the advantages provided, testbeds also pose some limitations. By nature, each testbed facility focuses on a specific area of interest (e.g. in IoT applications or M2M communication protocols) and therefore it's architecture and the services provided define the experiments supported. Other indicative limiting factors include the number and type of available resources and the number of simultaneous experiments the facility can support.

In order to overcome such limitations, experimenters have been working towards federating different testbeds. Such federated meta-testbeds enable different research groups to join forces towards diversifying and extending the existing experimental facilities. From these efforts emerged the prevailing paradigm of Testbed as a Service (TBaaS). According to this paradigm, the resources and the services of an individual facility are exposed to third parties via some RESTful APIs, over the Internet. This virtualization while obfuscating the underlying implementation details, enables an experimenter to utilize the facility while being agnostic of its complexity. Facilities that are virtualized by complying with commonly understood APIs can then be federated under the umbrella of a web-service. Hence, experimenters are provided with a single point of entry towards several facilities. Depending on the federation architecture, experimenters are able to provision resources for their experiment's which are provided by different individual facilities.

Our contribution. The existing experimental infrastructures mainly focus on resources that are statically deployed or characterized by low dynamics. For instance, they either refer to regular computer networks or to IoT infrastructure deployed within a static context, such as a smart building. In this work we present a holistic architecture for TBaaS, implemented in the context of IoT Lab European research project. The architecture has been adopted by the IoT Lab platform that can be accessed at `http://www.iotlab.eu/`. Through the IoT Lab one can access the resources of sensor testbeds from the Universities of Patras, Geneva, Surrey, from Mandat International and from the Ekonet mobile testbed. In addition resources of smartphone sensors can be accessed. The IoT Lab architecture enables us in particular to:

- Extend the range of resources by considering crowdsourced resources provided by the general public.
- Use a novel, generic yet specialized experiment mechanism. It allows a smart combination, composition and execution of diverse experiments to the users of the architecture.
- Adopt abstraction mechanisms that leverage devices, such as smartphones and smart wearables, as a distributed experimental infrastructure.

Related work. Internet of Things facilities cover a large number of topics from purely technical issues (e.g. routing protocols, semantic queries), to a mix of technical and societal issues (security, privacy, usability), as well as social and business themes ([14] and [15]). Federation of such facilities can be feasible with tools such as those introduced in [9]. The OneLab experimental facility, presented in [13], is a leading prototype for a flexible federation of testbeds that is open to the current Internet. GENI, the Global Environment for Networking Innovation [10], is a distributed virtual laboratory for transformative, at scale experiments in network science, services, and security. The Fed4FIRE federation framework [2] is gradually enabling experiments that combine facilities from the different FIRE research communities. Last but not least, the GEANT World Testbed Facility [12] focuses on regular computer networks.

Related crowdsourcing and crowdsensing platforms such as [1], [3] and [7] solely focus on mobile resources as a source of sensing data. Therefore they do not include any data annotation, or any other data coming from the knowledge of the crowd. Other platforms like EpiCollect [5] and PhoneLab [17] introduce the crowdsourcing concept. However they do not integrate other types of resources, such as testbed resources nor do they include any user profiling through which they can filter the crowd, or provide support mechanisms for incentives.

2 Resource Handling

Due to the increasing amount of electronic devices and sensors that are available, either by portable devices or through testbeds and experimental facilities, there has emerged a need to migrate all the available resources under the same umbrella. This migration allows an easy interaction between end-users and experimenters, on the one hand and available resources on the other. Key requirements in our architecture design were the federation of heterogeneous resources (e.g. static, mobile, portable, and crowd- sourced resources), the scalability of architecture in terms of mobile users and IoT integration and the simultaneous handling of a large number of resources and data during the experiment execution

2.1 Resource Description

Our architecture is gathering heterogeneous resources provided either by testbeds facilities, or by crowdsourcing (e.g. by the end-users of a facility or even the general public). In order to overcome complex migration problems with the heterogeneity of the resources, we adopted the RSpec (resource specifications) scheme [4]. RSpec was used for network related resources and had to be adapted for use by IoT ones. The RSpec, is an XML schema used by the resource providers in order to describe all the available resources in the architecture. This schema is simple, yet powerful, and includes all the necessary information to describe the resources adequately. The RSpec was mainly used for network resources and for this reason we had to expand its capabilities so that they would fit in our architecture requirements.

RSpec provides tags that describe several properties of each resource such as an IP address, a protocol for communication, an access port, or a location. In particular, the aforementioned tags are aligned with the types and function sets defined by the IPSO Application Framework. For instance, a luminance sensor following the IPSO Application Framework is categorized as "ipso.sen.lum". All resource providers generate an RSpec compliant XML file that is aggregated to a single architecture-wide description of available resources in the Resource Directory.

The schema provides tags that describe nodes (<node> </node>) which include properties allowing the system to directly access the resources of each node. These properties include the IP address (ip), the protocol the node understands (protocol) and the port (port).

Inside the <node> tag, the schema provides tags for individual resources (<resource> </resource>) that describe in detail the relative path that must be used by the architecture in order to request values from each resource, as well as the type of the resource (e.g. sensor or actuator). Inside the <resource> tag, the schema describes the resource using tags that follow the types and function sets defined by the IPSO Application Framework.

Other information that is contained inside the <node> tag includes an <interface> tag that provides more information about the component ID and a <location> tag that provides information about the physical location of the node. In the link below, we provide a snippet from an indicative RSpec XML file that describes some nodes in the Geneva's testbed. [8]

Through RSpec we can also describe RESTful web services that add functionalities to testbed resources. For example, we can convert a pulse meter to an energy one, make a temperature IoT resource from a weather API, or even provide alarm and notification services. For example, the resources available in the architecture are described by each resource provider using RSpec and can include all the necessary information to describe the resources needed to compose an experiment.

2.2 Diverse Resource Types

Static IoT resources. Each testbed which may be comprised of actuator motes, and either wireless, fixed or mobile sensors, provides them as resources to the platform. Each resource has a specific URL which invokes an API call. The resources of our testbeds follow a RESTful implementation via which GET, PUT or POST methods can be used to access them. Typical examples of such resources are the TelosB [11], Z1 [19] and Arduino IoT [8] devices.

Mobile/portable IoT resources. In addition to static IoT resources, mobile and portable testbeds can also be integrated. By doing so, we create networks of moving resources with multiple sensors that are capable of providing data and properties of temporal, technological, and spatial diversity. Existence of this type of testbeds provides the users of the architecture with more control over the choice of environment within which they want to deploy their experiments.

Virtual/modelled resources. Virtual resources are simulated nodes that act identically to the IoT resources and are running the same executable. The

[8] `http://129.194.70.52:8111/ero2proxy/service/type/xml_rspec`

difference to the physical ones comes from the way the reported values are generated. Those values are estimated by either taking into account other resources in the virtual environment only, or they can be interpolated by the physical resources of the same provider. The number of virtual resources that are deployed in each testbed side is fixed and set by the owner of the testbed. The functionality to add and integrate additional (virtual) modelled resources is also provided.

Crowdsourced Resources - Opportunistic/Participatory sensing. As presented in [16], each embedded sensor of a smart electronic device (e.g. smartphones), can be categorized as inertial, positioning, or ambient. Combining these categories, we can identify and measure the acceleration and rotational forces of a solid object, as well as measure the physical position of a device and various environmental parameters. The collection of measurement can be opportunistic or participatory. Opportunistic sensing takes place in the background, without needing the users to interact and have an active participation. Alternatively, participatory sensing urges users to be involved and provide the needed information, or data (such as scanning a QR code for localization purposes or answering a questionaire).

3 Experiment Composition and Execution

The experimenter is provided with a list of available resources that can view and reserve for their experiment. After the experimenter chooses and reserves the desired resources, he/she is prompted to the experiment composition module. In the background, the same RSpec XML schema is used to transfer the information regarding the resources reserved between the reservation module and the experiment composition module along with some meta-information on the experiment itself; e.g. duration and period of execution, human readable description of the experiment, etc. This information is incorporated in the RSpec document via tags such as the <research_id> tag, that provides the id of the parent research of the experiment to be composed, the <experiment_title> tag which provides the title the experimenter has given to the experiment to be composed and the <experiment_desc> which provides a short description of the experiment.

3.1 Experiment Composition

The experiment composition module receives this information and provides a simple but powerful mechanism with which the experimenter can define the details of how resources will be used in the context of "If This Then That" (IFTTT) scenarios. The final experiment consists of a set of these scenarios. The experiment composition module allows the experimenter to set the following actions:

Get a value from specified resources. The frequency of the reading request is set in minutes or hours and include one or more resources. The resources must be of type "sensor" and must be included in the experiment before the experimenter enters the main composition module. This action is called "reading".

As an example, a reading can be “Get a value from sensor 1 every 5 minutes between these 2 dates and times”.

Set a condition. A condition can be the average, absolute, minimum or maximum value of one or more resources being greater, equal or lesser than a set value. In the case of multiple resources a logical operator can be set. An example of a condition can be “The maximum value of sensor 1 OR the maximum value of sensor 2 to be greater than 5”.

Set an outcome. An outcome is an action that can be taken. This action is either to take more measurements from sensors or to actuate an actuator. Outcomes also include a logical operator in case there is more than one conditions. An example of an outcome could be “Actuate actuator 1, if all conditions are met (with logical AND)”.

Define an action. Actions are combinations of conditions and outcomes. Actions are set in an ”IF-THEN” form in order to clarify their meaning. An example of an action can be “IF condition 1 AND condition 2 are true THEN perform outcome 1”. The logical operator AND is actually defined in the outcome and not in the conditions, as specified above.

After the experiment scenario has been defined, it is dispatched to the execution module. The scenario is described in an XML schema called Experiment Description XML schema (ED XML). The Experiment Description XML defines a parent tag <experiment> </experiment> that encloses all other elements. The <measurements> tag defines the measurements database server information along with the <ip> and <port> sub-tags inside it. The next tag is a random identifier tag. This is generated during the ED creation randomly and is used to uniquely identify the experiment description. The tag that provides this identifier is the <identifier> tag.

Readings are included in the <reading> tag. Inside this tag, a <frequency> tag with a ”unit” property defines the frequency of the reading while <start> and <end> tags define the start and end of the readings period for the specified reading. The <resources> tag then defines which resources have to be probed for a reading every time it’s needed. These are defined using <id> tags that include properties ”component”, ”resource_id”, ”port”, ”ip”, ”protocol” and ”path”. The combination of these properties allow the execution engine to identify and reach the resources directly.

Actions are defined using the <action> tag. These include <conditions> and <outcome> tags. The <conditions> tag include the aggregation and logical operations as a tag and property respectively (e.g. <average logic=”and”>). Inside this tag, the resources are defined using an <id> tag and also the threshold is defined using a <threshold> tag. The <outcome> tag includes a property for the logical operator and inside the tag, resources are defined (either sensors or actuators) using <id> tags as above. An example of an ED XML is shown in Listing 1.1 in the Appendix.

3.2 Experiment Execution

When an experimenter finalizes the definition of an experiment at the Experiment Composition module, an Experiment Description XML document is created which is transferred to the Experiment Execution module which proceeds in parsing it and finding all necessary information in order to start running the experiment.

At first, the research ID, the experiment title and the experiment description are identified and posted as a new 'research' entity in the Resource Directory database. As already described, the Experiment Description XML document contains a number of readings and action tags. Each of these tags will spawn a thread to handle their tasks. A queue and two objects are used to handle communication between the readings and the actions. The *readingObject* notifies that a new measurement was taken. The *finishedObject* denotes that a reading thread was terminated.

Each reading tag has several resources with their contact information and a frequency with which they are to be read. Every one of those readings, spawns a new *getMeasurements*() thread tasked with obtaining the measurements from the resources in the time and with the frequency specified by the experimenter. The thread sleeps until it is time to take a measurement. When the measurement comes, the thread will wake up and call each resource associated with it for a measurement. After the measurements are taken the thread puts a *readingObject* to the queue and proceeds to sleep until the time comes to take a new measurement. When the time to finish the readings comes, the thread puts a *finishedObject* in the queue and then terminates.

Inside the actions tag there are a number of tied conditions and outcomes. Their information is parsed and summarized in two lists: one for the conditions called *conditionsList* and one for the outcomes called *outcomeList*. Then a thread for a function called conditionChecker(), with the two aforementioned lists as parameters is spawned. This thread reads the queue responsible for the communication between readings and actions. When a *readingObject* is read, it will evaluate the logic of *conditionsList* as specified in the Experiment Description XML. If it is evaluated to 'True', then the outcomes from *outcomeList* will be executed. When a *finishedObject* is seen, the number of the aforementioned threads will decrease by one. The thread will run as long as there are any *getMeasurements*() threads active.

3.3 Crowd Interactions

Crowd interactions require inputs from the smartphone users through surveys and questionnaires (Figure 1). The process of filtering and selecting the user in order to engage him/her in the specific research includes the following mechanisms available through the architecture: survey queries, survey lists and geofencing.

Survey Queries: A query is a mechanism that allows the experimenter to filter crowd users in a meaningful way in order to select the users needed

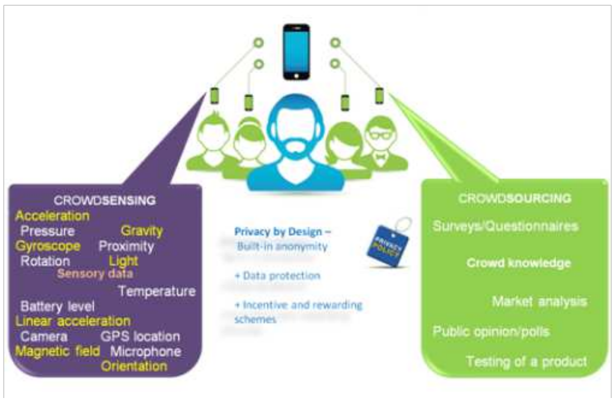

Fig. 1. Crowd participation in TBaaS architecture.

for the post of a mobile query. The filtering function is based on the socio-economic profile of the user which they voluntarily include during anonymous registration through the mobile app. The query is defined and then saved in the experimenter's profile so that it can be easily reused in the future, which makes it a very powerful tool as the crowd users constantly change in number throughout the architecture's lifetime. Queries, although static themselves, provide dynamic results in the form of sets of users that fit the set criteria.

Survey Lists: Every time a query is used, an up-to-date list of crowd users that meet the query's criteria is presented. The experimenter then has the opportunity to select individual recipients to form a survey list. A survey list is a static list of survey recipients that is used to send a survey to the mobile devices recipients. The content of the user list is anonymous and only social and economic data are associated with each entry. When the final survey list is compiled, it is saved under the experimenter's profile and can be used as the destination list in which to post a survey. A special case of a survey recipient list is the "all users" static list which includes all available users of the architecture.

Geofencing: Geofencing refers to the experimentation activity in which it is possible to setup a virtual perimeter on a real world geographic area and utilize this perimeter for determining if a mobile resource enters the area defined by the perimeter, exits such an area or is located inside or outside this area. This could be achieved, for example through the use of the GPS sensors, which are usually available on modern smartphones.

4 Engaging the Crowd

Contrary to traditional sensing systems that are designed specifically to monitor and collect data from fixed positions in their immediate environment and whose behavior can thus be engineered, in mobile crowdsensing systems each sensing point is controlled by a person that needs to give his/her consent in order for its device to participate in the system. This consequently introduces a high degree of unpredictability and unreliability to the system and thus raises demands for incentive and reputation mechanisms to engage the device owners by taking into account their individual preferences and behavior.

4.1 Incentives framework

To support the envisioned incentive models, apart from the experimenters and the regular users, new types of users were introduced:

– *Sponsors* can be individuals, companies or institutions following experiments and backing up the ones they find interesting.
– *Charities* are organizations or causes that the crowd can support through the allocation of their incentives.

A list of functionalities for the incentives and reputation framework has been specified to support the model as the most applicable to the architecture. Indicatively, the functionalities of this framework need to enable:

– Sponsors to back a specific research, specifying the amount of their contribution, which is transferred to the architecture and allocated to the research.
– Triggering of payments either periodically or when the research is completed, notifying users of the credits they have accumulated during their participation period.
– The user to have up-to-date information regarding their contribution(s), and the overall accumulated information about the credits for each research participation.

4.2 Reputation mechanisms

Another mechanism to motivate the crowd to participate in the research process is a reputation scoring scheme that provide users with information and statistics about the whole architecture as well as their part in it. The reputation mechanisms monitor user activities and calculate the their rating in a semi-automatic way.

The core module of the reputation mechanisms is a set of ranking functions that calculate the rank and the reputation of the experimenters, the experiments, the users, the devices and the platform itself. There are different types of ranking functions that adjust the rates with negative/positive contributions, i.e. with a five star rating scale or with a flag functionality that characterizes a user or an experiment (e.g. blocking a user). These functions run automatically in the back-end of the architecture and calculate the rates taking into account some statistics about the usage of the architecture, on one hand, and the rate that the users give through the mobile crowdsourcing tool and the experimenters through the website portal, on the other.

The ranking functions that calculate automatically the rates for experimenters take into account the statistics of each experimenter (e.g. the number of his completed experiments, if he provides reports with results from experiments, etc.), the evaluation of users for the experimenter's experiments and the rank of his ideas for proposed experiments. The rate of users that participate in experiments via the mobile crowdsourcing tool is calculated by functions based on the participation of each user (e.g. since when he is using the architecture, his response rate to experiments, the resources he provides for experiments, etc.) and the rate of their devices.

5 Architecture Scalability

In order to evaluate the performance, an extensive analysis of different non-functional properties of the architecture has been conducted (scalability, reliability and availability). This section describes an extensive scalability study, where we have identified three scenarios with different demands in terms of network bandwidth and analyzed the overall performance of the system. Figure 2 depicts the architecture's network architecture with all of its components (application, testbeds and TBaaS server).

We evaluate our architecture on the IoT Lab platform that is running on a server hosted in a Swiss data center providing a bandwidth of 100 Mbps (symmetrical, no SLA). For this study three scenarios with different demands in terms of both sensing and sourcing data were identified, as follows:

- *Use Case 1:* High-end scenario: sensing every 10 seconds and sourcing every minute.
- *Use Case 2:* Average scenario: sensing every 1 minute and sourcing every 30 minutes
- *Use Case 3:* Low-end scenario: sensing every 5 minutes and sourcing once daily

In regards to bandwidth requirements, use cases one and two can be characterized as high-end and average scenarios since they require constant environmental monitoring as to not disrupt the buildings usage whereas the third

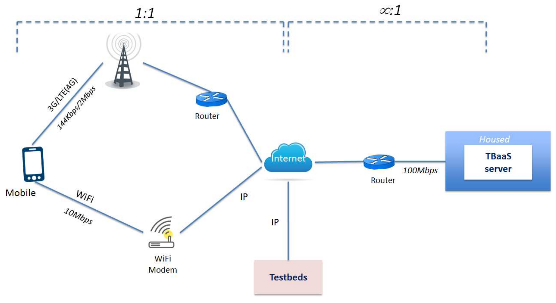

Fig. 2. architecture's network setting.

use case can be characterized as a low scenario since it doesn't require online responsiveness. For each of the scenarios, the following packages and sizes are considered:

– Sensing package: The sensing package is related to the message sent by the smartphone containing sensing observations (accelerometer, GPS, luminosity, humidity, temperature, etc.) or sensed values coming from a testbed resource (thermometer, humidity sensor, light sensor, noise sensor, etc.). In the mobiles case, we use 5 Kb as packet size having measured on the phone messages averaging 1,15Kbs of size, whereas for the testbed part, we consider the message size of 1 Kb, having measured an average size of 400 bytes for this type of messages;
– Sourcing package: The sourcing package is related typically to answering a questionnaire. Each question's package size is measured around 20 Kbs and each questionnaire consists of around 5 questions or 100 Kbs in total.

Table 1 shows the required bandwidth for each of the scenarios and the number of connections with the server at 100% and 50% of its capacity. For the mobile part, two bandwidths are calculated: the first using 3G/LTE for communication (1 to 1 communication) and the second using Wi-Fi with 20 connections to the same hotspot as a plausible and "safe" number. The 3G and 4G/LTE average bandwidths are around 0,5 Mbps and 2 Mbps to 12 Mbps.

Table 2 shows that for the high-end scenario the server can handle 8M or 4M connections at 100% or 50% of its capacity respectively. For the average scenario 48M or 24M connections and finally 240M or 120M connections for the low-end scenario. Considering the average scenario with the server with capacity at 50%, we can have up to 24M testbed resources connected and communicating their values to the architecture.

6 Practical Applications

Testbed as a Service offers the capability to conduct researches involving both crowd and IoT interactions. The following three scenarios showcase the capabilities and range of scenarios feasible by the architecture.

Table 1. Mobile Side - Bandwidth requirements, number of connections with server.

Mobile Side	BW (3G/LTE Kbps)	BW (Wi-Fi)	Capacity 100%	Capacity 50%
High-end scenario	0,271	5,417	369.231	184.615
Average scenario	0,017	0,347	5.760.000	2.880.000
Low-end scenario	0,002	0,045	44.883.117	22.441.558

Table 2. Testbed Side - Bandwidth requirements, number of connections with server

Testbed Side	Bandwidth (Kbps)	Capacity at 100%	Capacity at 50%
High-end scenario	0,013	8.000.000	4.000.000
Average scenario	0,002	48.000.000	24.000.000
Low-end scenario	0,0004	240.000.000	120.000.000

Light control scenario for a building management system [6]. The end goals are to monitor the energy consumption, to automate the lighting and to save energy. It uses static and crowd lent IoT devices together with surveys as a way to learn the crowd's opinion. The first step is to monitor the energy consumption. Then a group of crowd users based on their geolocation is created and a message is sent, informing them about the experiment and their role in it. The research requires passive light measurements from their sensors as well as opportunistic ones for their location within the building. These values determine whether or not the lights will be turned on. Questionnaires forwarded via the architecture determine the user's satisfaction and the need to readjust the parameters of the experiment.

Environmental monitoring scenario [18]. It's a cross disciplinary scenario which involves monitoring indoor and outdoor environmental data and correlating them with the crowd happiness. It uses both the crowd's opinion as well as IoT resources. Interaction with the crowd is realized through surveys. A collection of the crowd geolocation data at the time of posting the survey is necessary, in order to relate the survey responses with the geolocalized environmental data obtained from testbeds like ekoNET [18]. ekoNET is a network of mobile IoT sensor devices capable of monitoring temperature, humidity, pressure and air quality. These sensing data are also tied with a location measurement. Over a period of time some correlation between the gathered sensor data and the crowd's opinion may appear. To keep user participation high, proper incentivization is essential.

Virtual and modelled resources scenario. The objective as with first research is to run an energy efficiency scenario. The traditional way of doing so is by deploying static IoT devices tasked with measuring the luminance level and based upon their readings actuating the lights. With the help of virtual resources these measurement points can be augmented with virtual sensors. To do so the values produced by the sensors in the outer boundary of the building are used to create a dataset of external light data along with timestamps in order to identify patterns of external light coming into the rooms. In this fashion the number of physical resources is decreased. To run this scenario prior knowledge of the position of the static sensors in the building is required.

7 Conclusions

In this paper we presented a service based architecture for IoT testbeds which exposes the operations of a testbed as services, enables diverse facilities to be federated in a scalable and standardized way and enables the seamless integration of crowdsourced resources. The architecture enables an experimenter to access a diverse set of resources and orchestrate experiments via a common interface. Moving forward we plan to increase the control and capabilities of the experimenters and integrate additional devices and functionalities to the experiment composition module and present real world applications of the platform.

Acknowledgments. This work was supported by the EU/FIRE IoT Lab project - STREP ICT-610477

References

1. APISENSE - Crowd-sensing made easy! `www.apisense.com/`. retrieved April 2016.
2. Fed4FIRE project. `http://www.fed4fire.eu/`. retrieved April 2016.
3. funf - Open sensing framework. `http://funf.org/`. retrieved April 2016.
4. Rspec, fed4fire project. `http://fed4fire-testbeds.ilabt.iminds.be/asciidoc/rspec.html`. retrieved April 2016.
5. D. M. Aanensen, D. M. Huntley, E. J. Feil, F. al Own, and B. G. Spratt. EpiCollect: Linking smartphones to web applications for epidemiology, ecology and community data collection. *PLoS ONE*, 4(9):e6968, 09 2009.
6. C. Angelopoulos, O. Evangelatos, S. Nikoletseas, T. Raptis, J. Rolim, and K. Veroutis. A user-enabled testbed architecture with mobile crowdsensing support for smart, green buildings. In *Communications (ICC), 2015 IEEE International Conference on*, pages 573–578, June 2015.
7. C. M. Angelopoulos, S. Nikoletseas, T. P. Raptis, and J. Rolim. Design and evaluation of characteristic incentive mechanisms in mobile crowdsensing systems. *Simulation Modelling Practice and Theory*, 55:95 – 106, 2015.
8. Arduino. Arduino motes. `https://www.arduino.cc/`. retrieved April 2016.
9. J. Aug, T. Parmentelat, N. Turro, S. Avakian, L. Baron, M. A. Larabi, M. Y. Rahman, T. Friedman, and S. Fdida. Tools to foster a global federation of testbeds. *Computer Networks*, 2014. Special issue on Future Internet Testbeds.
10. M. Berman, J. S. Chase, L. Landweber, A. Nakao, M. Ott, D. Raychaudhuri, R. Ricci, and I. Seskar. GENI: A federated testbed for innovative network experiments. *Computer Networks*, 61:5 – 23, 2014. Special issue on Future Internet Testbeds Part I.
11. Crossbow. Telosb. `www.willow.co.uk/TelosB_Datasheet.pdf`. retr. April 2016.
12. F. Farina, P. Szegedi, and J. Sobieski. GEANT world testbed facility: Federated and distributed testbeds as a service facility of GEANT. In *Teletraffic Congress (ITC), 2014 26th International*, pages 1–6, Sept 2014.
13. S. Fdida, T. Friedman, and T. Parmentelat. *New Network Architectures: The Path to the Future Internet*, chapter OneLab: An Open Federated Facility for Experimentally Driven Future Internet Research, pages 141–152. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.
14. A. Gluhak, S. Krco, M. Nati, D. Pfisterer, N. Mitton, and T. Razafindralambo. A survey on facilities for experimental internet of things research. *Communications Magazine, IEEE*, 49(11):58–67, November 2011.
15. J. Horneber and A. Hergenroder. A survey on testbeds and experimentation environments for wireless sensor networks. *Communications Surveys Tutorials, IEEE*, 16(4):1820–1838, Fourthquarter 2014.
16. S. A. Hoseini-Tabatabaei, A. Gluhak, and R. Tafazolli. A survey on smartphone-based systems for opportunistic user context recognition. *ACM Comput. Surv.*, 45(3):27:1–27:51, July 2013.
17. A. Nandugudi, A. Maiti, T. Ki, F. Bulut, M. Demirbas, T. Kosar, C. Qiao, S. Y. Ko, and G. Challen. PhoneLab: A large programmable smartphone testbed. In *Proceedings of First International Workshop on Sensing and Big Data Mining*, SENSEMINE'13, pages 4:1–4:6, New York, NY, USA, 2013. ACM.

18. B. Pokric, S. Krco, D. Drajic, M. Pokric, I. Jokic, and M. Stojanovic. ekoNET - Environmental monitoring using low-cost sensors for detecting gases, particulate matter, and meteorological parameters. In *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2014 Eighth International Conference on*, 2014.
19. Zolertia. Zolertia motes. `http://zolertia.io/`. retrieved April 2016.

Appendix A

In the following Listing 1.1 it is shown an Experiment Description XML example. In this example, a reading is requested between two specified date-times to be taken every 1 minute from a resource. The experiment also defines that if the average value of one of these resources is less than 1 the light control defined must be actuated. All measurements recorded through experiments are stored in the MongoDB measurements database. This means that experiments can also be conducted without even defining conditions and actions, if what is needed is only data from specific sensors to be taken.

Listing 1.1. Experiment Description XML

```xml
<?xml version='1.0' encoding='utf-8'?>
<experiment>
  <measurements><ip>129.194.70.52</ip><port>9000</port></measurements>
  <identifier>IemNuXCQTGasLMo5mMjkqxPYKewJYhkh</identifier>
  <reading>
    <frequency unit='minutes'>1</frequency>
    <start>2015-06-19 14:54</start>
    <end>2015-06-19 14:54</end>
    <resources>
      <id component=urn:publicid:IDN+iotlab:mitestbed:mitestbed+node+
      node7.mitestbed' resource_id='undefined' port='61616'
      ip='2001:620:607:5800:0:0:0:1c' protocol='coap'
      type='sensor' path='/co2' unit='ppm'></id>
    </resources>
  </reading>
  <action>
    <conditions>
      <average logic='and'>
        <id component='urn:publicid:IDN+iotlab:unigetestbed:unigetestbed+node+
        C3S7A1-LightLevel' resource_id='undefined' port='8111' ip='129.194.70.52'
        protocol='http' type='sensor' path='/lum' unit='lx'></id>
        <threshold type='less' value='100'></threshold>
      </average>
    </conditions>
    <outcome logic='and'>
      <id component='urn:publicid:IDN+iotlab:ctitestbed:ctitestbed+node+
      node_light_control' port='568' unit='none' resource_id='undefined'
      ip='2001:620:607:5f00::15' protocol='coap' type='actuator' path='PUT-dev0-1'></id>
    </outcome>
  </action>
</experiment>
```