iHome: Secure Smart Home Management in the Cloud and the Fog

Euripides G.M. PETRAKIS, George MYRIZAKIS
School of Electrical and Computer Engineering, Technical University of Crete (TUC), Chania, Crete, Greece

Abstract. iHome is a smart home management service that runs in the cloud. The service addresses the need of users to monitor and control their homes remotely provided that the home devices are “smart” themselves (i.e. they can be connected to the internet and operated remotely). Home devices transmit their identifier, measurements and status to a fog node and from there to the cloud. To mitigate concerns in regards to data protection, response time and communication delays in delivering large amounts of data to the cloud, all services for the home are realized within a fog node installed at home. This information becomes available to registered users in the cloud based on subscriptions (i.e. to users authorized to review and respond to this information). User access rights are defined based on user roles (i.e. cloud administrators, home moderators and residents). Besides data publication and subscription services, an innovative feature of iHome, is a rule-based event management service which forwards alerts to subscribed users for responding to critical events (i.e. incidents of fire, malfunctioning appliances at home). iHome is implemented based on principles of Service Oriented Architecture design as a composition of RESTful services.

Keywords. smart home, service oriented architecture, event, authorization, security, cloud computing, Internet of Things

1. Introduction

Smart homes are the building blocks of smart cities. E-government, Green development and Energy Management policies for large residential areas may be benefit from the evolution smart home technologies. Smart policies and solutions at smart home level may influence policies applying at a larger scale (i.e. smart city level) and the reverse: Smart city policies may drive the development of smart home solutions for more efficient resource management (e.g. water, energy, internet etc.). Engineering and architectural solutions for smart homes are reviewed in [9][14].
The idea of Internet of Things (IoT) combined with Cloud Computing, opens new horizons in the field of smart home technology. Smart home systems are now designed as two-fold solutions involving two interoperable and interconnected components: The IoT side (front-end) to support the network of home devices and their connection with the Internet (via a gateway or edge node) and, the cloud side (back-end), where home control functionality is implemented and data are collected for permanent storage and analysis.

The use of sensors installed in smart appliances and their capability for Internet connectivity provides significant benefits in applications areas that require fast and continuous monitoring from anywhere. In real-life smart home and smart city applications, huge amounts of data are collected and analyzed. Improved home management solutions or better business policies can be designed based on the results of this analysis. These solutions have to be scalable (to deal with the ever-increasing number of smart homes and users, and finally, with the increased size of data), cost-effective, respond within reasonable time (e.g. considering the time constraints of each application) and, address concerns related to security, users privacy and data safety. While smart home solutions are deployed for the home and the cloud, security concerns arise at both places. Although cloud systems are considered to be more secure for deploying IoT applications, users and data are exposed to many risks as IoT at homes operates in the periphery of the cloud, it is open to many users and, as such, it is generally less protected than the cloud itself. The security concerns for the front-end part are addressed in [20]. In order to guarantee security at the back-end, data must be transferred and stored securely to the back-end, user activities and system operations must be monitored at all times and, access to data and services must be allowed based on user roles and authorization [18].

The cloud is the ideal environment for implementing the back-end component of smart home applications due to reasons related to its affordability (no up-front investment, low operation costs), ease of deployment (taking advantage of IaaS and PaaS solutions already available by many cloud vendors), low maintenance costs (i.e. easy software updates and enhancements), scalability (i.e. computational resources can be added on demand) and accessibility (i.e. smart home services can be accessed anytime from anywhere over the Web).

Cloud computing has some inherent disadvantages and limitations. Even though the cloud may offer virtually unlimited computational resources, internet bandwidth may impede application performance or, the cost of transferring and processing large amounts of data in the cloud, call for hosting smart home services at the front-end side (rather than hosting all of them at the back-end) closer to the place where data is generated (e.g. a server operating at home). The distance between front and back-end is also the main reason for long delays in responding to critical events occurring at home. To address these limitations, the paradigm of Fog Computing emerged lately, starting from Cisco [4]. Fog computing can be assumed as an extension of cloud computing, bringing virtualized services closer to the edge of the network (e.g. closer to IoT devices). Fog brings the benefits of low latency (due to its close proximity with the IoT devices), location awareness and increased security, privacy and availability. Ef-
forts to standardize architectures and platforms for extending the cloud to support functional edge nodes are currently underway [11].

The concept of smart home is interesting and trendy. Existing solutions relying on appliances connected to the Internet are now becoming available as commercial services, supporting functionalities ranging from home automation, intelligent home control and security (e.g. anti-burglary systems), to ambient or assisted living and health monitoring. In addition, off-the-shelve smart devices with the desired functionality are currently becoming available in the market at affordable prices and can be integrated into smart home applications. An important feature of all systems (and also of iHome) is user interaction with a mobile device and intelligent user interfaces (using e.g. voice interfaces) over the Internet. Commercial applications promise fast responses, high availability and, (most important) high reliability. iHome in not intended to compete commercial solutions in terms of performance but rather, to show how a cost effective smart home system based on innovative services can be designed and deployed in the cloud and fog using well established, open-source technologies and principles of service oriented design.

iHome, is a service that runs in the cloud and the fog and implements a typical smart home scenario: Smart devices installed at home can transmit measurements or status information to gateways and from there to the cloud. To mitigate concerns in regards to data security and in regards to delays in delivering large amounts of data to the cloud, services for the home are realized within a fog node installed at home. Access to this information is allowed only for users registered to iHome in the cloud based on their role and authorization. (i.e. cloud administrators, home administrators, residents). Permission to access homes is granted by cloud administrators. Home users (i.e. home administrators and residents) may access home information based on subscriptions. Besides measurements, alerts are a special type of information which is generated when events take place. These can be simple events (e.g. room temperature exceeds a threshold) or critical events (e.g. incidents of security breach or fire). The events are described in terms of rules involving measurements and threshold values.

iHome solution, is based on smart devices for the home IoT (front-end or fog) and on a composition of RESTful micro-services for the cloud side (back-end). Because a large number of smart devices for the front-end is not available to us, we decided to rely on software simulating the operation of smart devices at homes. This allowed us to shift focus from device specific functionality (e.g. IoT transmission protocols and vendor specific device functionality), to the design and implementation of the front and back-end solutions. Inspired by the concept of Service Oriented Architectures (SOA) [7], the iHome solution makes use of modular services implementing fundamental functionalities communicating with each other and offering important benefits, such as scalability, re-usability multi-tenancy, increased accessibility and security through powerful APIs. We run an exhaustive set of experiments using simulated (but realistic) data aiming

---

3https://thinkmobiles.com/blog/best-smart-home-apps/
to evaluate both, iHome response time and scalability. The results demonstrate that the system can handle up to a large number of users and data in real time.

The rest of this paper is organized as follows: Issues related system design and architecture are discussed in Sec. 3. Future trends and challenges in the design and implementation of smart home solutions are discussed in Sec. 4. iHome implementation along with the REST interfaces of its component services are presented in Sec. 4. Evaluation results are presented in Sec. 5 followed by conclusions and issues for future research in Sec. 6.

2. From Smart Homes to Smart Cities: Vision and Research Challenges

The impact of a smart home application is not limited to home boundaries. It relates to many IoT use cases with great economic impact. Smart cities [1] are realized as collections of smart homes at many scales (e.g. neighborhoods, communities, urban and rural areas) that are connected with each other and with the cloud where data are transferred for analysis. The analysis of history (log) data collected from a large number of homes (eventually this information becomes big) can lead to important conclusions in relation to people habits and behavior, their needs, the causes of events or, in relation to energy consumption at peak and non-peak hours. The analysis may provide the means for policy makers, energy managers to improve their plans for more efficient, safer and profitable management of a smart city.

Deployment and operation costs appear to be the barriers standing in the way of commercialization of smart home solutions (with another one being the maturity of the technology), but it is also seen as the greatest driver for it. The basic challenge that drives work in smart home systems should address, relates to the lack of technological components to aid and simplify the development of cross-domain applications (e.g. smart-home, smart-city) in a fast and secure way. Among these, security, openness, interoperability through advanced connectivity and usability (i.e. applicability) issues, play an important role in deploying interconnected solutions. Technology advances mainly in security, IoT communications and IoT systems interoperability are the driving forces that will guide development of cross domain applications that are open, secure and scalable.

2.1. Smart Home Use Cases

The value of smart home use cases is discussed next followed by a discussion on research challenges related to implementing the smart city scenario.

Energy Saving [14] can lead to reduced energy consumption in smart cities by improving the energy efficiency of buildings. The scenario requires detecting energy consumption patterns in a city population in order to propose effective resource (e.g. energy, water, food) and cost saving plans by avoiding unnecessary consumption. The idea is to stimulate and motivate users to adapt their behaviour in order to contribute to energy and cost savings. Social Networks and serious game platforms [6] are the technology means by which users can participate in the scenario. Smart home residents are evaluated based on the actions taken (related to energy consumption). Eventually, the platform will allow citizens to understand the daily trade-offs in delivering energy to a single household.
Ambient Intelligence aims at facilitating and enhancing human interaction with their living environment. Sensors embedded at homes (e.g., in smart furniture or appliances) can detect users' actions. In this scenario, the application incorporates different types of information such as living habits (e.g., time seated, ergonomic adaptation of sitting surface to body problems or difficulties) and relation to environmental aspects (e.g., lighting, temperature), interaction with TV, etc. The analysis of this information will allow companies to incorporate new features into their products, opening a range of new possibilities for the production of smart products.

Health monitoring has great societal and economic impact. Its purpose is to provide assistance to sensitive user groups with emphasis to the elderly, disabled, patients, and also children, remotely. Wearable devices (e.g., a smart wristband that captures vital signs such as body temperature, pulse rate, respiration rate, blood pressure (in relation to the monitoring of body activity and environmental values), provide vital information about the user's health status and activity. The sensing system might integrate information from more sources (e.g., motion and visual sensors) in order to assess the condition of the user and detect possible emergency situations of users at risk and whose condition requires prompt assistance.

2.2. Research Challenges

Technology advances in IoT connectivity and security will enable design and implementation of next generation smart city use cases that operate in a reliable, efficient, and secure way.

2.2.1. Smart Home Connectivity

Considering the large number of interconnected devices and the complex nature of smart city infrastructures, IoT connectivity relies mostly on wireless protocols. Emerging infrastructure technologies for supporting wireless connectivity in smart cities include narrow-band IoT such as Low Power Wide Area Networks (LPWAN) and 5G. Choosing a connectivity solution depends on the needs of an application.

5G is widely considered as an enabling technology for the Internet of Things. It is designed to support use cases with bandwidth demanding requirements, millisecond latencies, ultra-high network reliability and massive connection density. 5G will not only support mobile telephony but also services through net slicing: Different bands of the spectrum are assigned to support services in different application domains (e.g., remote health monitoring and personalized medicine in the narrow band). Telecom providers can operate services such as Software Defined Networks (SDN) and establish Virtual Private Network (VPN) services specifically for IoT providing increased security of the communications. If a smart city scenario requires wireless transmission of big data volumes (e.g., video surveillance deployment with large numbers of cameras), 5G is the most appropriate connectivity solution. For the e-Health scenario in particular, Theummler et al. focus on infrastructure requirements considering patient monitoring parameters, the nature of data (e.g., images and multimedia),
user interactions (e.g. between patients and health providers) outside the hospital.

For short-range connectivity, Bluetooth\(^4\) and Bluetooth Low Energy (BLE) can connect IoT devices (e.g. sensors) to a gateway and then to the cloud via some backhaul (e.g. a cellular network and the internet). However, the number of IoT devices (sensors) that can be connected to the gateway is limited (especially for sensors and gateways running on battery and unless the gateway is connected to a sustainable power source). Bluetooth and BLE, the same as WiFi and Zigbee, are not suited for long-range performance, while cellular networks are costly, consume a lot of power, and are expensive. For use cases with low data traffic (e.g. monitoring of the living environment), Low Power Wide Area Network (LPWAN) technologies \(^{19,26}\) are most suitable connectivity solution for smart city IoT projects.

LPWAN technologies fill the gap between cellular (e.g. 3G, 4G, LTE) and short-range wireless (e.g. Bluetooth, WiFi and ZigBee) networks. The trade-off is the achievable data and error rate (i.e. LPWAN protocols do not guarantee delivery of data packets). LPWAN technology falls short in terms of QoS compared to cellular standards and this means that an operator cannot use it to provide the kind of Service Level Agreements (SLAs) that are critical to the customers of fast paced applications requiring high speed collection of large data volumes. However, LPWAN specifications fits well the requirements of many IoT applications (e.g. smart cities, home automation, industrial automation, environmental monitoring, e-Health) who need to transmit to gateways small data quantities periodically over a long range, while maintaining long battery life (e.g. a wearable health sensor that only transmits when vital measurements exceed some predefined threshold).

LPWAN networks are being deployed now because the cost to deploy the network in unlicensed bands requires much less capital than the cost of its 3G, 4G counterparts. SigFox\(^5\) and LoRa\(^6\) are the main competitors in this landscape. The same do other technologies such as Weightless, Ingenu, NB-IoT, and LTE-M\(^7\) which enable long rage devices to be connected to telecommunication networks with the latter two been standardized within 3G and 4G respectively. Typically, an IoT application can be deployed only if the network is already there. However, for vendors that need to deploy IoT applications on their own and run the network by themselves, LoRa is a good option. LoRa supports 2-5Km ranges in urban areas (entire cities can be covered with a few LoRa antennas) and up to 15Km in rural areas. It works in the unlicensed spectrum below 1Ghz which comes at no cost. It is an asynchronous protocol, which is optimal for battery lifetime and cost. There are no royalty issues with LoRa (except of the LoRA modulation chip which is produced by Semtech\(^8\)).

Leveraging 5G capabilities (i.e. ubiquity, integrated security and network management) a network of LoRa devices can be developed anywhere, without

\(^{4}\)https://www.bluetooth.com
\(^{5}\)https://www.sigfox.com/
\(^{6}\)https://www.lora-alliance.org/
\(^{7}\)https://www.link-labs.com/lpwan
\(^{8}\)https://www.semtech.com
installation of additional network equipment and without the need for network management (which is offered by the cellular network). 5G technology comes with its own authentication, authorization and accounting framework thus minimizing the effort for supporting this functionality in LPWAN networks. 5G slices are secured by deploying security as Virtual Network Functions (VNFs) in the different network slices and chaining the security VNF, as specified in the global network security policy [15]. Soon, telecom providers will support LPWAN functionality and synergy with 5G.

2.2.2. Smart Home Security

A smart home operates at the periphery of the network, so data and devices are exposed to the internet and to access by unauthorized or malicious users [10]. Securing IoT applications that are distributed over several IoT and cloud infrastructures is a challenging task. New security issues emerge also due to the fact that IoT nodes may be untrusted.

Regulatory entities recommend that the principles of security by design and security and privacy by default should be applied to IoT. The Security Framework [9] by Industrial Internet Consortium (IIC) highlights the need for monitoring devices, networks and applications at the edge of the network and the cloud [17]. The security pillar of the OpenFog reference architecture [11] specifies the FaaS (Fog-as-a-Service) security monitoring functionality for devices at network end. Data protection is crucial in all application domains. EU’s GDPR [11] has a significant impact on IoT systems design. In e-Health in particular, a potential intrusion may not only lead to vulnerable systems (e.g. susceptible to personal data theft) but also lead to risks in human life (e.g. disruption of important sensors monitoring patients etc.). Anonymization, pseudonymization and data protection techniques can also be applied to avoid exposing data to unauthorized third parties. Distributed and hybrid identity and authorization management must be applied (e.g. un-authorized access to data and services is protected by OAuth2.0 [12]).

Due to the size and complexity of modern IoT systems, security threats can be detected in many aspects of system operation and relate mainly to malicious users’ behavior detection which is expressed as (a) fraud detection in which case, authorized of unauthorized users operate the system for the purpose of unfair or unlawful gain or, (b) intrusion, in which case, unauthorized users are attempting to disrupt normal system operation. Similar behavior can be detected in virtualized environments such as the environment of a cloud provider (now affecting the operation of the system in scale and a large number of users) with certain economic and operational impact [8].

A key problem for IoT applications that collect large amounts of data, is the on-the-fly and real-time solutions for anomaly detection (e.g. system failure, se-

---

10https://www.iiconsortium.org/pdf/IIC_PUB_G4_V1.00_PB.pdf
12https://oauth.net/2/
curity breaches), for improving Quality of Service (QoS) or for detecting security leaks and vulnerabilities. Traditional large-scale solutions for malicious behavior or malfunction detection, suggest either continuous monitoring of the state of fog or cloud components or periodically monitoring of system logs, or both. In large-scale cloud and fog-distributed infrastructures, system logs are becoming big data as time passes. The monitoring of large system typically logs resorts to retrospective analysis of big data [17].

3. iHome Design and architecture

iHome is a novel Future Internet (FI) service for data collection from IoT devices in an automatic, generalized and modular way. Building upon principles of Service Oriented Architectures (SOA) design and driven by the key requirements of today’s IoT systems for adaptability, low-cost and scalability, iHome architecture is modular and expandable. iHome is implemented in OpenStack and FIWARE[13], an open-source distributed cloud infrastructure funded by the European Union (EU). iHome encompasses IoT-A [23] design principles in an attempt to develop an innovative IoT platform that supports generic services and IoT devices (i.e. independent of connectivity and not coupled to specific IoT protocols). IoT-A proposes an Architecture Reference Model (ARM) defining the principles and guidelines for generating IoT architectures, providing the means to connect vertically closed systems both, at the communication layer (i.e. where IoT devices interact with the system) and the service layer (i.e. where services are provided).

We followed a valid design approach that identified the functional and non-functional requirements of the system and specifically, (a) the functional components and their interaction, (b) the information that is managed and how this information is acquired, transmitted, stored and analysed, (c) the physical software entities that support the functional and information activities, (d) the requirements for assuring data, network and user security and privacy. iHome (reference) architecture is described by a set of UML diagrams [16] including (a) information (class) diagram describing information that is handled by the system, (b) deployment diagram showing interaction of system components, (c) activity diagram describing flowcharts for several types of user actions the most important of them being, system login request (user authentication and authorization), request for new account, request to access a device, event handling and, (d) an architecture diagram.

3.1. Class Diagram

The class diagram describes the structure of the main entities (classes) of the application domain (i.e. the home system), their properties, operations allowed on these entities and their interrelationships. Typically, it is represented as a class (IS_A) hierarchy with the most general classes at the top and more specific classes (i.e. specializations of general classes) lower in the hierarchy. Other types of relationships between classes (i.e. “object properties”) can be also defined (e.g. “creates”, “inserts” property relationships). A class is also described by a set of

\[13\text{https://www.fiware.org}\]
attributes (i.e. “data properties”) together with a set of operations that can be executed on entities of this class. Entities lower in a class hierarchy inherit all properties and operations of entities higher in the hierarchy. The UML diagram of Fig. 1 describes all entity types (classes), their properties, their relationships as well as the operations allowed on each class. There are different types (classes) of users that interact with the system. Each type of user is associated with different services that handle different types (classes) of information.

**User:** Describes users interacting with the system. Each user is described by an identifier (ID), email, and a name being displayed. This class is associated with methods for importing and deleting users. The class is specialized
in three user sub-categories (sub-classes) namely, Cloud Moderator, Home Moderator, and Residents.

**Cloud Moderators (Infrastructure Administrators):** Except their competence to providing cloud services, they are responsible for providing services to users subscribing to iHome. In particular, they are responsible for performing Create, Read, Update, Delete (CRUD) operations on (a) homes (i.e. they can register new homes to the system), (b) they can register new home users (i.e. home administrators and residents) and define their access rights (i.e. for associating residents and devices to homes or rooms).

**Home Moderators (Home Administrators):** They are the managers of the home: (a) They configure, maintain and monitor iHome services, (b) they are responsible for performing CRUD operations on rooms, devices and events, (c) they are responsible for granting access rights to residents and of monitoring their operations, (d) they have access to status information for rooms, home devices and events; (e) they have access to history (log) data, (f) they define rules for handling events.

**Residents:** They are entitled to accessing information pertinent to their role and perform read (only) operations on information regarding devices and rooms they are subscribed and based on access rights assigned to them by the home administrator.

**Home:** Describes home as a collection of rooms which in turn may contain monitoring and control devices. A home is associated with the home and the Cloud Moderator who are entitled to monitor and control the operation of the smart home (by adding or deleting residents and devices). One or more residents may subscribe to a home with some access rights.

**Room:** Describes the rooms of a home by the collection of devices associated with each room. The condition of a room is described by three attributes (i.e. temperature, humidity, luminosity). There can be up to nine different types of devices installed in a room.

**Device:** Devices are associated with rooms and are assigned to rooms by Home Moderators. Nine device types are defined (i.e. air condition, oven, refrigerator, washing machine, lamp, blinds, tv and alarm).

**Rule:** Defined conditions for triggering events at home. It is specialized to **Condition Rules** which trigger events when a condition is met (e.g. turn-on air-conditioning when room temperature exceeds a predefined threshold) and, **Time Rules** which trigger events periodically at specific times of the day (e.g. lock all doors at 10pm).

**Alert:** In cases of devices malfunctioning or in cases of events, a notification is sent to all users subscribing to this information.

### 3.2. Activity Diagrams

Activity diagrams define the workflows (i.e. sequence of operations) which are executed when an action take place. There are two types of actions allowed in a smart home: Actions initiated by users and actions triggered by rules. Later type workflows are realized as a rule-based service which forwards alerts to subscribed users in response to critical events (i.e. incidents of fire, malfunctioning
appliances at home). In turn, events are triggered when rules are violated (e.g. the temperature is not within the pre-specified value range). Actions initiated by users are distinguished by user type and their authorization. Fig. 2 summarizes all possible user actions and their workflows. The intention of a user to perform a particular type of action (e.g. add a device or display monitoring data) is expressed by issuing his/her preference on a Web application.

3.3. Architecture

iHome architecture is modular, expandable and generic (i.e. it can be adapted to diverse use case scenarios). The implementation relies on modular services that run in the cloud and implement fundamental functionalities. They offer important benefits such as, connectivity of smart devices, scalability, openness, re-usability, multi-tenancy, increased accessibility and security. Each module is designed as a RESTful service which is able to interact with other RESTful services over HTTPS. This design highlights significant advantages that include (in addition to the comprehensive structure and re-usability of services), easy replacement of modules and flexible system configuration in order to adapt to future needs of the application. Fig. 3 illustrates the iHome solution with its two system nodules namely, front-end and back-end respectively.

3.3.1. Front-end

In addition to being a gateway (whose role is to support real-time IoT data collection from home devices and their communication with the cloud) it implements additional services that transform the gateway to a fog node. In iHome set-up, a home is a composition of any number of rooms with devices installed in each room (i.e. each device is associated with a room). iHome implements also device simulators producing temperature, humidity and luminosity values. In addition, a room or device accepts change of mode of operation commands (i.e. on/off, heating/cooling etc.) and read status commands (i.e. normal or malfunction). The communication between services and between the front and back-end is implemented in HTTPS. The front-end is implemented as a composition of RESTful services:

**Data collector**: Accomodates any kind of sensor protocol (e.g. Bluetooth, Zigbee) that can be supported by smart devices and collects measurements from all home devices. It implements a two-way communication with home devices; it decodes and decrypts device physical payloads (i.e. physical devices transmit encrypted data); translates physical device payloads in JSON format; it transfers control commands received from authorized users (e.g. turn on/off commands, change mode of operation commands etc.) to specific devices. It is the only component that is affected by the property of a device to apply a specific protocol (e.g. Bluetooth). The rest of the system is protocol agnostic since all data are communicated and processed in JSON.

**Database (DB)**: It is a MongoDB storage service for device data and users. Device data are transmitted to the cloud at predefined time intervals (i.e. ev-
Figure 2.: iHome workflow diagram.
ery minute). In cases of Internet loss, data are cached and transmitted to the back-end when connection is up again. The identifiers and roles of authorized users (i.e. home moderators and residents) are stored as well. Each authorized user has received an OAuth2.0 key (“token”) together with an XACML file encoding user roles, which are generated in the back-end (see Sec. 3.3.2). The SSL certificate (i.e. asymmetric key) is also stored. It is used to establish an encrypted HTTPS connection between the front and the back-end and to create the session symmetric key of an HTTPS session. Each time a new HTTPS session between the front and the back-end is initiated, a new symmetric key is generated.

**Connectivity service:** It implements the HTTPS connection between the front-end and the cloud. Data forwarded to the cloud are encrypted using the symmetric SSL key (or decrypted upon receipt). Similarly, data received at the back-end are decrypted with the symmetric key (or encrypted prior to transmission).

**User authorization:** Services running at both system ends are protected by an OAuth2.0 mechanism. Users receive an OAuth2.0 token and an XACML file upon login in the back-end (see Sec. 3.3.2). If a user issues a request to access a service or device in the front-end, the OAuth2.0 token is added to the header of the request. User Authorization service uses the token to retrieve user’s identity and XACML file from the database. The request is executed (i.e. it is forwarded to the device or service specified in the request) only if the user has received the necessary authorization (this information is encoded in the XACML file); otherwise the request is aborted.

---

15https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml
16https://oauth.net/2/
3.3.2. Back-end

The back-end is a composition of autonomous RESTful services communicating with each other using HTTPS. Access to data and services is protected by an OAuth2.0 mechanism. Some service modules are available as reusable Generic Enablers (GEs) on FIWARE catalogue. All others are implemented in Java. iHome back-end implements the following services:

**Connectivity service**: In line with connectivity service running in the front-end, it implements secure communication between the front and back-end (decoded data are formatted in JSON).

**User identity management**: Implements (a) identification services for users and manages their profiles, (b) authorization services supporting access control based on user roles and access policies. It is provided by Keyrock Identify Management (IdM) service of FIWARE. Keyrock IdM offers Single Sign On (SSO) service for users and applications and implements an OAuth2.0 mechanism. Applications accessing protected resources are checked whether they are authorized to do so using their OAuth2.0 credentials. All users must be registered to FIWARE cloud and have a username and password in order to log into the application. Keyrock IdM issues a “session key” (token) encoding users’ identity and an XACML file encoding user’s authorization and roles.

**User authorization**: Although Keyrock IdM is entitled to issue a session key, it does not perform any authorization check by itself. This operation resorts to user authorization service. This service is typically invoked by application logic (see below) which is responsible for issuing requests to other services and for calling services in order. Every (user) request must have the session key in its header. Once a request reaches application logic and prior to invoking the target service, the session key is forwarded to authorization service. From there, a request is send to Keyrock IdM which responds with users’s identity information. User authorization service is entitled to approve or reject the request based information encoded in the XACML file. Application logic executes requests only if they are approved by this service.

**Publication and Subscription (PS) service**: It acts as a media broker (i.e. passing information to users subscribed to services). It mediates between devices publishing measurements or receiving control commands to specific devices. Each time a new sensor is registered to iHome, or a new measurement becomes available, this component is updated (i.e. a new entity is created in this service or its value is updated, respectively) and a notification is sent to all entities subscribed to this information. Subscribed users can also retrieve entity values from this service. This service is implemented using the Publish-Subscribe Context Broker service of FIWARE.

---

17https://oauth.net
18https://www.fiware.org/developers/catalogue
Event Processing (EP) service: The home moderator may create rules for handling events. All rules are stored in the cloud and are operated by the application logic service (see Sec. 3.3.2). For each rule, application logic service sends a subscription request to PS service. The role of PS service is to trigger the execution of the rule each time room environment measurements change (e.g. temperature, humidity, luminosity measurements in iHome scenario). If the conditions of the rule are met (e.g. the measurement exceeds the pre-defined threshold value), an alert is generated which is forwarded to application logic service which is responsible for handling the event. Home users subscribed to this event get a notification and, in cases of critical events, application logic service triggers an action (e.g. alarm sound or phone calls to authorities for handling the event in the case of fire or security breach at home). Fig. 4 illustrates the EP sequence diagram. In a future implementation, an EP service is foreseen for the home (i.e. the front-end) as well. The role of this EP service is to provide faster responses for events occurring at home. All rules defined at the back-end must be downloaded to the front-end or, two different sets of rules can be defined for front and back-end respectively.

![Event Processing (EP) sequence diagram](image)

Figure 4. Event Processing (EP) sequence diagram

Application logic: Orchestrates, controls and executes services in order. When a request is received, it is dispatched to the appropriate service. For example, services regarding user accounts and access rights are dispatched (through application logic) to user authorization service. It performs basic security controls (e.g. checking if a session between the front-end and the cloud or between the Web application and the cloud has been initiated). Threshold violations (reported by EP service) are managed by this component. It creates notifications when a rule triggers actions (e.g. alerts when a device is malfunctioning). Such notifications and alerts are forwarded to the Web application.
History database: It implements a data storage service for log data (i.e. device measurements, actions, events), users (i.e. user profiles, access rights), rules triggered by the event processing service each time a device value is updated. It is realized as a non-SQL database (i.e. a JSON storage service) using MongoDB. The NoSQL format of MongoDB handled semi and unstructured data (e.g. JSON formatted data) and adapts to the model of IoT data (does not apply a common tabular format for all data types). In general, and compared to relational databases (e.g. MySQL), NoSQL databases (such as MongoDB) are extremely performant and scalable (i.e. they can handle very large data sizes while maintaining very good performance) while supporting common operations such as key indexing, queries by individual keys, as well as non-common ones, such as text searching (e.g. MongoDB), and distributed storage (e.g. Cassandra).

Web application (UI): Implements the user interface and runs on a Web browser. A user can access the system by providing user name and password after he/she registers to FIWARE cloud. This request is forwarded to Keyrock IdM which issues the OAuth2.0 token (session key). This token is added to the header of every request made by the user to iHome. The request is approved or denied based on user role and authorization. The Web application is implemented using HTML and CSS/jQuery.

Data analytics: This component is particularly useful as data become big. The big data processing service handles analytics for useful feedback for improving system responses, enhancing users experience and applying better business plans. It is described here for completeness of the discussion but it is left as future work.

4. iHome Implementation

iHome implements the front and back-end services of Sec. 3.3 in FIWARE using Java-EE, the Java-based platform for server programming which includes a Java API for RESTful Web services and JSON processing. All service is implemented as RESTful APIs communicating with HTTPS. Individual services or groups of services are deployed in the same or different Virtual Machines (VMs). iHome is deployed in three Virtual Machines (VMs) in different clouds. The first two VMs are deployed in the FIWARE node of TUC (FIWARE is a federation of distributed cloud infrastructures running OpenStack). These are small flavor VMs with one virtual processor (x86_64 processor architecture, 2,800 Mhz, first level cache size 32 KB, second layer 4,096 KB cache size), 2,048 MB RAM, 20 GB hard drive capacity. Each VM runs Ubuntu 18.04 and Tomcat server. The first VM runs application logic, user authorization, Event Processing, database and connectivity service. The second VM runs Publication and Subscription service alone. Finally, the third VM runs Keyrock IdM service which is provided by a shared VM installed in a remote FIWARE node (e.g. in Spain). All VMs may re-

\(^{21}\)https://www.mongodb.com
\(^{22}\)http://cassandra.apache.org
\(^{23}\)https://www.fiware.org
\(^{24}\)https://www.openstack.org
receive up to a large number of simultaneous requests as the number of users and requests increase.

Network delays are expected due to the nature of this design. However, the experimental results (Sec. 5) demonstrate that iHome is capable for responding in real time even under heavy workloads. Nonetheless, the advantage of this Service Oriented Architecture (SOA) is, system modular design, ease of configuration that best suits the need of the application, ease of maintenance and expandability. For example, more services can be added at-run time or, any service can be moved to a different VM (on the same or different cloud) with minimum overhead (i.e. only the IP of the service will change).

iHome can be viewed as an ensemble of REST services communicating with each other. Once the REST interface of a service is known, other services can communicate with it using HTTPS. Therefore, the way to describe the implementation of iHome is by means of the REST interfaces of its component services.

Cloud moderator services: Cloud moderators are entitled to handle other users (i.e. home moderators and residents) and their homes. They are entitled to create, update and delete users. Table 1 is the REST interface of the service. Table 2 is the REST interface of the service for handling homes. Cloud moderators are entitled to create new homes, update and delete existing ones.

<table>
<thead>
<tr>
<th>Method</th>
<th>URL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>POST</td>
<td>/superusers</td>
<td>Create new superuser</td>
</tr>
<tr>
<td>GET</td>
<td>/superusers</td>
<td>View all superusers</td>
</tr>
<tr>
<td>GET</td>
<td>/superusers/{superuserId}</td>
<td>View superuser with id={superuserId}</td>
</tr>
<tr>
<td>PUT</td>
<td>/superusers/{superuserId}</td>
<td>Edit superuser with id={superuserId}</td>
</tr>
<tr>
<td>DELETE</td>
<td>/superusers/{superuserId}</td>
<td>Delete superuser with id={superuserId}</td>
</tr>
<tr>
<td>POST</td>
<td>/users</td>
<td>Create new user</td>
</tr>
<tr>
<td>GET</td>
<td>/users</td>
<td>View all users</td>
</tr>
<tr>
<td>GET</td>
<td>/users/{userId}</td>
<td>View user with id={userId}</td>
</tr>
<tr>
<td>PUT</td>
<td>/users/{userId}</td>
<td>Edit user with id={userId}</td>
</tr>
<tr>
<td>DELETE</td>
<td>/users/{userId}</td>
<td>Delete user with id={userId}</td>
</tr>
</tbody>
</table>

Table 1. REST API for handling home moderators (super users) and residents (users).

<table>
<thead>
<tr>
<th>Method</th>
<th>URL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>POST</td>
<td>/homes</td>
<td>Create new home</td>
</tr>
<tr>
<td>GET</td>
<td>/homes</td>
<td>View all homes</td>
</tr>
<tr>
<td>GET</td>
<td>/homes/{homeId}</td>
<td>View home with id={homeId}</td>
</tr>
<tr>
<td>PUT</td>
<td>/homes/{homeId}</td>
<td>Edit home with id={homeId}</td>
</tr>
<tr>
<td>DELETE</td>
<td>/homes/{homeId}</td>
<td>Delete home with id={homeId}</td>
</tr>
</tbody>
</table>

Table 2. REST API for handling homes.
Home moderator services: The following services are available to home moderators. These implement functionality for the management of residents, rooms, devices and also rules. Table 3 is the interface for operations on rooms. Home moderators are entitled to create, update, monitor and operate devices installed in a home. Table 4 is the REST interface of this service. They can create rules for handling events as of Table 5. If the conditions of a rule are violated, an event is triggered (i.e. the event service of Fig. 4 is invoked). Finally, home moderators have control over residents (i.e. they are allowed to monitor their operations and update their access rights to rooms and devices). Table 6 is the REST interface of this service.

<table>
<thead>
<tr>
<th>Method</th>
<th>URL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>POST</td>
<td>/homes/{homeld}/rooms</td>
<td>Create new room in home with id = homeld</td>
</tr>
<tr>
<td>GET</td>
<td>/homes/{homeld}/rooms</td>
<td>View rooms of home with id={homeld}</td>
</tr>
<tr>
<td>GET</td>
<td>/homes/{homeld}/rooms/{roomId}</td>
<td>View room with id={roomId} of home with id={homeld}</td>
</tr>
<tr>
<td>PUT</td>
<td>/homes/{homeld}/rooms/{roomId}</td>
<td>Edit room with id={roomId} of home with id={homeld}</td>
</tr>
<tr>
<td>DELETE</td>
<td>/homes/{homeld}/rooms/{roomId}</td>
<td>Delete room with id={roomId} of home with id={homeld}</td>
</tr>
</tbody>
</table>

Table 3. REST API for handling rooms.

<table>
<thead>
<tr>
<th>Method</th>
<th>URL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>POST</td>
<td>/homes/{homeld}/devices</td>
<td>Insert new device in home with id={homeld}</td>
</tr>
<tr>
<td>GET</td>
<td>/homes/{homeld}/devices</td>
<td>View devices of home with id={homeld}</td>
</tr>
<tr>
<td>GET</td>
<td>/homes/{homeld}/rooms/{roomId}/</td>
<td>View devices of room with id={roomId} that belongs to home with id={homeld}</td>
</tr>
<tr>
<td>GET</td>
<td>/homes/{homeld}/rooms/{roomId}/devices/{deviceId}</td>
<td>View device of room with id={roomId} that belongs to home with id={homeld}</td>
</tr>
<tr>
<td>PUT</td>
<td>/homes/{homeld}/rooms/{roomId}/devices/{deviceId}</td>
<td>Operate device of room with id={roomId} that belongs to home with id={homeld}</td>
</tr>
<tr>
<td>DELETE</td>
<td>/homes/{homeld}/devices/{deviceId}</td>
<td>Delete device of home with id={homeld}</td>
</tr>
</tbody>
</table>

Table 4. REST API for handling devices.

Services for residents: Residents have no control over other users. They can only view monitoring information and operate devices for which they are granted access by home moderators. Table 7 is the REST interface of this service.
### Table 5. REST API for handling rules.

<table>
<thead>
<tr>
<th>Method</th>
<th>URL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>POST</td>
<td>/homes/(homeld)/rules</td>
<td>Create rule in home with id=homeld</td>
</tr>
<tr>
<td>GET</td>
<td>/homes/(homeld)/rules</td>
<td>View rules of home with id=homeld</td>
</tr>
<tr>
<td>GET</td>
<td>/homes/(homeld)/rules/(ruleId)</td>
<td>View rule with id=ruleId of home with id=homeld</td>
</tr>
<tr>
<td>DELETE</td>
<td>/homes/(homeld)/rules/(ruleId)</td>
<td>Delete rule with id=ruleId of home with id=homeld</td>
</tr>
</tbody>
</table>

### Table 6. REST interface for handling residents.

<table>
<thead>
<tr>
<th>Method</th>
<th>URL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/homes/(homeld)/users</td>
<td>View users of home with id=homeld</td>
</tr>
<tr>
<td>GET</td>
<td>/homes/(homeld)/users/(userId)</td>
<td>View user with id=userId of home with id=homeld</td>
</tr>
<tr>
<td>PUT</td>
<td>/homes/(homeld)/users/(userId)</td>
<td>Edit access rights of user with id=userId that belongs to home with id=homeld</td>
</tr>
</tbody>
</table>

### Table 7. REST API for accessing home devices.

<table>
<thead>
<tr>
<th>Method</th>
<th>URL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/homes/(homeld)/devices</td>
<td>View devices of home with id=homeld</td>
</tr>
<tr>
<td>GET</td>
<td>/homes/(homeld)/rooms/(roomId)/devices</td>
<td>View devices of room with id=roomId that belongs to home with id=homeld</td>
</tr>
<tr>
<td>GET</td>
<td>/homes/(homeld)/rooms/(roomId)/devices</td>
<td>View device of room with id=roomId that belongs to home with id=homeld</td>
</tr>
<tr>
<td>PUT</td>
<td>/homes/(homeld)/rooms/(roomId)/devices/(deviceId)</td>
<td>Operate device of room with id=roomId that belongs to home with id=homeld</td>
</tr>
</tbody>
</table>

**User identification and authorization service:** Access to iHome is granted by the Keyrock Identity Management (IdM) GE of FIWARE. After successful login to FIWARE cloud, Keyrock IdM creates a session with the user identifier based on user’s role and authorization (i.e. the user receives an access token encoding his/her access rights). Table 8 illustrates the REST interface to this service.

**Publication and Subscription (PS) Service:** This service establishes a communication channel between services (or users and services) based on subscriptions. Every time data is created or updated, users and services subscribing to this information get notified. This is realized by creating context entities in PS service which are constantly monitored for value changes (i.e. users subscribe to context entities and receive alerts whenever value changes occur). The notification is in the form of a REST request sent by the service to a URL. Table 9 is the REST interface of PS service that allows to create and update monitoring entries (i.e. entries capable for generating subscription events). In iHome, each device is as-
Table 8. REST interface of Keyrock IdM service.

<table>
<thead>
<tr>
<th>Method</th>
<th>URL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/oauth2/authorize?response_type=code&amp;client_id={clientId}&amp;state=xyz&amp;redirect_uri={redirectURI}</td>
<td>Request authorization code from keyrock. Client_id is provided by fware upon application registration. Redirect URI is defined by the application creator and it points to the URI that the code will be sent</td>
</tr>
<tr>
<td>POST</td>
<td>/oauth2/token</td>
<td>Request access token from Keyrock</td>
</tr>
<tr>
<td>GET</td>
<td>/user/access_token={accessToken}</td>
<td>Get user information by sending the accessToken that we received from Keyrock</td>
</tr>
</tbody>
</table>

Table 9. REST API of Publication and Subscription service.

<table>
<thead>
<tr>
<th>Method</th>
<th>URL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>POST</td>
<td>/v2/entities</td>
<td>Create entity</td>
</tr>
<tr>
<td>GET</td>
<td>/v2/entities</td>
<td>Retrieve all entities</td>
</tr>
<tr>
<td>GET</td>
<td>/v2/entities/{entityId}</td>
<td>Retrieve entity with id={entityId}</td>
</tr>
<tr>
<td>PUT/PATCH</td>
<td>/v2/entities/{entityId}</td>
<td>Update entity with id={entityId}</td>
</tr>
<tr>
<td>DELETE</td>
<td>/v2/entities/{entityId}</td>
<td>Delete entity with id={entityId}</td>
</tr>
<tr>
<td>GET</td>
<td>/v2/entities/{entityId}/attrs/{attrName}</td>
<td>Retrieve value of attribute with name={attrName} of entity with id={entityId}</td>
</tr>
<tr>
<td>PUT/PATCH</td>
<td>/v2/entities/{entityId}/attrs/{attrName}</td>
<td>Update value of attribute with name={attrName} of entity with id={entityId}</td>
</tr>
<tr>
<td>DELETE</td>
<td>/v2/entities/{entityId}/attrs/{attrName}</td>
<td>Delete value of attribute with name={attrName} of entity with id={entityId}</td>
</tr>
</tbody>
</table>

Event Processing (EP) service: Entities in PS service are associated with information about their value and status, as well as with rules defined on thresholds or time. The role of PS service is to trigger the execution of the rule each time room environment measurements change (e.g. temperature, humidity, luminosity measurements in iHome scenario). In turn, events are triggered when values of PS entries exceed some predefined thresholds. Alternatively, events are generated periodically at the time specified. The EP service exposes a REST interface as of Table [11].
Table 10. REST API for managing subscriptions.

<table>
<thead>
<tr>
<th>Method</th>
<th>URL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>POST</td>
<td>/v2/subscriptions</td>
<td>Create subscription</td>
</tr>
<tr>
<td>GET</td>
<td>/v2/subscriptions</td>
<td>Retrieve all subscriptions</td>
</tr>
<tr>
<td>GET</td>
<td>/v2/subscriptions/{subId}</td>
<td>Retrieve subscription with id=subId</td>
</tr>
<tr>
<td>DELETE</td>
<td>/v2/subscriptions/{subId}</td>
<td>Delete subscription with id=subId</td>
</tr>
</tbody>
</table>

Table 11. REST API for Event Processing service.

<table>
<thead>
<tr>
<th>Method</th>
<th>URL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>POST</td>
<td>/eventHandler</td>
<td>Notifications from Publish/Subscribe service are send here.</td>
</tr>
<tr>
<td>POST</td>
<td>/timeEvent</td>
<td>Create time rule</td>
</tr>
</tbody>
</table>

Storage service: Anonymized history data (i.e. user access data, events, monitoring data) are stored in JSON format to the cloud. Data collected can be analyzed for revealing regularities or hidden associations in data. These, in turn, may reveal system bottlenecks, malfunctions, user preferences or user habits which, can be taken in consideration in future system updates improving system operations (e.g. leading to enhanced users experience or better business decisions). Table 12 is the REST interface to this service.

Table 12. REST API for handling the history Database.

<table>
<thead>
<tr>
<th>Method</th>
<th>URL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>POST</td>
<td>/logs/{logType}</td>
<td>Create log entry. LogType can be: homeLog, userLog, roomLog, deviceLog</td>
</tr>
<tr>
<td>POST</td>
<td>/alert</td>
<td>Create alert when an alarm triggers</td>
</tr>
<tr>
<td>POST</td>
<td>/notifications/rule</td>
<td>Create notification when a rule is executed</td>
</tr>
<tr>
<td>POST</td>
<td>/notifications/deviceStatus</td>
<td>Create notification when a device malfunctions</td>
</tr>
</tbody>
</table>

5. iHome Evaluation

We run an exhaustive set of experiments and we analyze the performance limits of the services running in the cloud. We study also system scalability (i.e. how system response time increases with the number of connected users). For each experiment we report average response time (over 2,000 requests) for the most common operations. Each operation can be executed either sequentially (i.e. one after the other) or in parallel. We use ApacheBench\(^\text{25}\) (Apache HTTP

\(^{25}\)https://httpd.apache.org/docs/2.4/programs/ab.html
server benchmarking tool) to send multiple simultaneous requests to iHome. In ApacheBench we are opted to define the total number of requests and how many of them will be executed simultaneously. We report average response time for 2,000 requests executed in a sequence (concurrency = 1) and also for increasing values of concurrency (i.e. concurrency > 1). All service requests address the user authorization and Keyrock IdM service at the back-end which checks if the user or service issuing the request has the necessary access rights. All measurements of time below account also for the time spent for the communication between VMs or between services within the same or different VMs. Table 1 summarizes the performance of the following basic operations executed in iHome.

A **home posts room values** to the connectivity service (at the back-end) and published in PS service (i.e. the corresponding entities are updated) and the database. Users subscribing to this information get a notification about this change on the Web application (through application logic service). The EP service subscribes always to this information and triggers the execution of rules that govern the monitoring of a home. If these values are within the pre-defined limits, no action is taken; otherwise (e.g. temperature exceeds a limit), EP service notifies the home moderator to take action.

A **user requests room values** from a home. The request is issued on the Web application. From there, it is forwarded to PS service which always holds the most recent values of all home entities. Finally, the Web application receives the requested information (again through application logic which is responsible for orchestrating operations running in iHome).

**Insert a device at a home.** The home moderator needs to start monitoring a new device in a home. The request is issued on the Web Application and its forwarded to application logic service. This operation will insert a new entity in PS service with the three default values (i.e. temperature, humidity and luminosity). The request is forwarded (through connectivity service) to the front-end. After decoding, user authorization service in the front-end will check if the user has the necessary access rights. If access is granted, the new device is created (data collector and database services are updated as well). The device is published at PS service (i.e. a new entity is created) and the home moderator can define users subscribing to the new PS entity. The user receives a notification upon successful completion of this process.

Response times improve with the simultaneous execution of requests (i.e. the Apache HTTP server switches to multitasking) reaching their lowest values for concurrency between 50 and 150. Even with concurrency = 300 the average execution time per request is close to real-time and resource usage is well below 100% in most cases.

iHome may produce big amounts of data and requests, requiring large processing capabilities which can surpass the capacities that our experimental system set-up is able to provide. In this iHome set-up, core services are implemented in three VMs. Presumably, overloading the VMs might occur in an application with a much larger number of concurrent users. An obvious solution to dealing with performance would be to employee additional VMs each running
Table 13. Performance of basic iHome services

| A home posts room values to the cloud: POST localhost:8228/getRoomValues |
|---------------------------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|
| Concurency | 1 | 50 | 100 | 150 | 200 | 300 |
| Time (ms)  | 88.29 | 67.23 | 63.29 | 66.22 | 70.29 | 77.71 |
| CPU (%)    | 97.94 | 99.43 | 98.86 | 98.14 | 96.24 | 99.39 |
| RAM(%)     | 69.67 | 75.65 | 80.96 | 87.10 | 93.96 | 95.88 |

| Get room values from a home: GET localhost:8448/homes/homeId/rooms/roomId |
|---------------------------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|
| Concurency | 1 | 50 | 100 | 150 | 200 | 300 |
| Time (ms)  | 67.22 | 61.91 | 63.93 | 65.83 | 67.29 | 73.59 |
| CPU (%)    | 36.16 | 74.96 | 78.26 | 77.66 | 73.08 | 68.63 |
| RAM(%)     | 73.29 | 75.52 | 77.39 | 80.51 | 83.03 | 84.75 |

| Insert a new device at a home: POST localhost:8448/homes/homeId/devices |
|---------------------------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|
| Concurency | 1 | 50 | 100 | 150 | 200 | 300 |
| Time (ms)  | 370.5 | 100.1 | 101.3 | 103.1 | 104.4 | 109.8 |
| CPU (%)    | 21.84 | 58.68 | 53.72 | 51.65 | 53.02 | 49.87 |
| RAM(%)     | 66.43 | 68.96 | 70.74 | 72.86 | 74.90 | 77.76 |

a single service (or a small group of services). Alongside, we can allocate additional VMs implementing the same service (or groups of services) thus having more than one VM sharing the load.

6. Conclusions and future work

Leveraging PaaS functionality we show how a smart home management system is designed and implemented as a service in the cloud. iHome exhibits a highly modular SOA design based on micro-services running both, at a home (i.e. a fog node) and in the cloud allowing users to monitor and control their homes remotely in real-time. iHome has been tested in a realistic scenario with up to three hundred concurrent users. The experimental results reveal that, iHome is capable of responding in real time even under heavy workloads. Besides being fast and modular, iHome innovative design relies on secure cloud services permitting access to services and information based on access rights and authorization (i.e. all services are protected by a security mechanism). At the same time, iHome implements secure communication for data and services over HTTPS. As a future extension, iHome must be tested using a large number of physical devices and appliances installed at homes. Implementation of data analytics functionality in a public cloud, in a use case with many homes connected to iHome, is also an interesting direction for future work.

Acknowledgement

The ideas for smart city use cases and research challenges are inspired by teamwork in preparing the 1IoT and ADON project proposals submitted for funding in “Horizon 2020” program of the EU. We are grateful to many excellent partners across the EU for their cooperation.
References


the Software to Data paradigm by Utilizing Hybrid Cloud Technology. In *IEEE Symposium on Bioinformatics and Bioengineering (BIBE’2013)*, pages 1–6, Chania, Greece, November 2013.


Index of Terms

- Smart Home
- Smart City
- Internet of Things (IoT)
- Cloud Computing
- Security
- Infrastructure as a Service (IaaS)
- Platform as a Service (PaaS)
- Edge Computing
- Authorization
- Micro-service
- Ambient Intelligence
- Health monitoring
- Bluetooth
- Bluetooth Low Energy (BLE)
- Low Power Wide Area Network (LPWAN)
- Service Level Agreements (SLAs)
- SigFox
- Weightless
- Ingenu
- NB-IoT
- LTE-M
- Lora
- Virtual Network Function (VNF)
- Quality of Service (QoS)
- OpenStack
- IoT-A
- Unified Modeling Language (UML)
- Generic Enabler (GE)
- Gateway
- Zigbee
- Single Sign On (SSO)
- Keyrock Identity Management (IdM)
- Web Application
- History Database
- MongoDB
- NoSQL database
- OAuth2.0
- jQuery
- FIWARE
- Web Service
- JSON
- REST
- HTTPS
- Virtual Machine (VM)
- Tomcat Server
- Event Processing
September 2019

- Publication and Subscription
- Service Oriented Architecture (SOA)
- eXtensible Access Control Markup Language (XACML)

Acronyms

- IoT: Internet of Things
- REST: REpresentational State Transfer
- PaaS: Platform as a Service
- IaaS: Infrastructure as a Service
- SOA: Service Oriented Architecture
- GE: Generic Enabler
- BLE: Bluetooth Low Energy
- HTTPS: Hypertext Transfer Protocol Secure
- SLA: Service Level Agreement
- NoSQL DB: Not only SQL Database
- GDPR: General Data Protection Regulation
- JSON: JavaScript Object Notation
- LPWAN: Low-Power Wide-Area Network
- UML: Unified Modeling Language
- FI: Future Internet
- PS: Publication and Subscription
- XACML: eXtensible Access Control Markup Language

Glossaries

- Internet of Things (IoT): The network of interconnected devices that collect and exchange data;
- Cloud Computing: The on-demand delivery of compute power, database, storage, applications, and network resources via the internet with pay-as-you-go pricing;
- REST: A software architectural style that defines a set of constraints to be used for creating Web services;
- PaaS: A set of cloud computing services available as a platform;
- IaaS: A set of high-level cloud computing services as a platform which are used to de-reference low-level physical computing resources;
- Edge Computing: A computing model that brings virtualized resources closer to the edge of the network;
- Service Oriented Architecture (SOA): A software development model based on the composition of independent and re-usable software components that communicate with each other through well defined interfaces by message passing;
- Generic Enabler (GE): Basic services which are used as building blocks for software development;
- BLE (Bluetooth Low Energy): A power conserving variant of Bluetooth;
September 2019

**Service Level Agreement (SLA):** A contract that specifies what the service provider is responsible for;

**eXtensible Access Control Markup Language (XACML):** A declarative language based on XML describing how to evaluate access requests according to rules;

**HTTPS:** An extension of the Hypertext Transfer Protocol (HTTP) for secure communication;

**General Data Protection Regulation (GDPR):** A legal framework that sets guidelines for the collection and processing of personal information of citizens of the European Union (EU);

**NoSQL Database (NoSQL DB):** A database mechanism that is not only modeled by means tabular relations;

**JavaScript Object Notation (JSON):** A lightweight format for storing and transporting data;

**Low Power Wide Area Network (LPWAN):** A wide area network designed for low power but long-range communications at a low bit rate;

**Unified Modeling Language (UML):** A modeling language by means of diagrams for specifying, visualizing and documenting software systems;

**Publication and Subscription (PS):** A messaging pattern allowing users or machines to publish information and, others to subscribe to it and receive it when it becomes available;

**Smart Home:** A home setup where appliances and devices are connected to the internet so that they can be controlled remotely;

**Smart City:** A city that incorporates technology solutions in order to enhance the quality and performance of city services (e.g. reduce resource consumption, transportation times and overall costs);

**Author biographies**

- Euripides G.M. Petrakis holds a Ph.D degree from University of Crete, Greece since 1993. He is a professor and director of the Intelligent Systems Laboratory at the School Electrical and Computer Engineering of TUC, Greece. His current research interests focus on Semantic Web, Software Engineering, cloud computing and IoT.

- George Myrizakis received a Diploma in Electrical and Computer Engineering from TUC, Greece in 2019. His research focuses on Software Engineering, IoT and Cloud Computing.