Journal of Parallel and Distributed Computing 132 (2019) 262–273

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

journal homepage: www.elsevier.com/locate/jpdc

A framework for real time end to end monitoring and big data oriented management of smart environments

Antonio Celesti *, Maria Fazio

Department of Mathematical, Computer Science, Physics and Earth Sciences - MIFT, University of Messina Viale F. Stagno d'Alcontres, 31 98166 Messina, Italy

HIGHLIGHTS

- Monitoring of IoT device requires the analysis of big data.
- We propose a piece framework integrating AllJoyn with MongoDB and Storm.
- Different data patterns were designed to manage several smart home scenarios.
- Simulation experiments prove the feasibility of the proposed system.

ARTICLE INFO

Article history:
Received 2 August 2017
Received in revised form 23 August 2018
Accepted 13 October 2018
Available online 22 November 2018

Keywords:
Cloud computing
Edge computing
IoT
Big data
Monitoring
Management

ABSTRACT

Nowadays, the success of Internet of Things (IoT) applications depends on the intelligence of tools and techniques that can monitor, manage, and verify the correct operations of smart ecosystems including sensors and big data analytics tools, typically deployed in Cloud and Edge computing datacenters. In this paper, we propose a framework for the monitoring and management of IoT system that integrates the AllJoyn functionalities, useful to interconnect IoT devices, MongoDB, to implement Big Data storage, and Storm, to run real-time data analytics. We implemented the proposed framework and we tested its main functionalities in a smart home application scenario. In our experimentation, we investigated three different data patterns, i.e., regular, event-based, and automated, in order to evaluate performance of our framework in terms of response time under different operational conditions. Experimental results show that the latency of the monitoring and service strongly depends on the type of management application running in the system, whereas it is lightly affected by the data patterns.

1. Introduction

Nowadays, the combination between Internet of Things (IoT), Cloud computing, and Edge computing technologies is pursuing new opportunities in delivering services, representing a strategic approach for ICT companies for increasing their business.

IoT is a very challenging paradigm that enables things, i.e., various embedded devices, to be connected anytime, any place, with anything and anyone ideally using any path/network and service. Integrating smart IoT systems with the Cloud and the Edge is currently the main objective of ICT operators because it enables the development of various user-driven services and applications connected to sensors and actuators. In fact, IoT applications depend on the intelligence of tools and techniques that can monitor, manage, and verify the correct operations of smart ecosystems including sensors and big data analytics tools, typically deployed in Cloud and Edge computing datacenters [16,18].

In order to support the needs of people for interacting with smart IoT environments, applications controlling many different embedded devices should be able to modify their behavior according to the state of the environment and customization requests. Such emerging environments, includes, for example, smart cities, smart mall, smart home, and so on. For example, a smart home includes multiple examples of IoT devices that produce big data and that have to be monitored and managed by means of particular data patterns. For example, (i) the curtains in a flat can be automatically moved according to the intensity of the external light; (ii) if the door of the fridge is open for a long time a notification can be sent to the user; (iii) from 14 July, 2017 to 18 July, 2017 the coffee machine has to be switched on at 6:45; (iv) the system can consult a table of energy cost and suggest to the user the best time when particular household electrical appliances should be used in order to save money; (iv) if different household electrical appliances are switched on and their energy consumption is closer to the

* Corresponding author.
E-mail addresses: acelesti@unime.it (A. Celesti), mfazio@unime.it (M. Fazio).

https://doi.org/10.1016/j.jpdc.2018.10.015
0743-7315/

allowed threshold of energy, the system sends a notification to the user in order to prevent a possible blackout. These are only a few possible cases of application, in fact there is no limit to the possible functionalities that can be developed. The system can be very complex if we consider rooms, homes/flats, and whole residence. At any level can be developed particular data patterns that allow an intelligent management of the involved environment.

In such a scenario, the monitoring and management of IoT devices is not trivial due to the huge amount of data that have to be stored and analyzed. In this paper, we propose a framework able to address these challenges, and in particular we refer to a smart home reference scenario, in order to discuss the effective applicability of our solution to real use cases [22]. The proposed framework makes use of consolidated technologies already used in different applications, that are AllJoyn, MongoDB and Storm respectively.

AllJoyn [10,28] is an open source project that provides a framework for distributed applications running across heterogeneous devices with an emphasis on networking, mobility, security, and dynamic configuration. The AllJoyn system enables to handle the problems arising from heterogeneous distributed systems in IoT environments. In a mobile environment, devices constantly join/leave the proximity of other devices. To this end, AllJoyn has been designed to implicitly support mechanisms of proximity and mobility. This leaves developers free to focus on the main functionalities that they want to build in their applications. MongoDB [20] is one of the major distributed document-oriented NoSQL databases, useful to store and efficiently manage Big Data originated by physical and composed measurements on the environment and on systems. Apache Storm [2] is a distributed system for real-time processing of data streams, that can be adopted to process video and audio data coming from the environment and crossrelating them with additional information [15].

The framework proposed in this paper is able to integrate different functionalities of the above technologies in order to provide an innovative solution to monitor and manage IoT systems. The main advantage of the proposed framework is the ability to quickly process sensed and historical data and streams, exploiting IoT devices and embedded and computing systems at the edge of the IoT network.

We tested the developed framework in a smart home case of study. This type of application scenario is a good choice for us to easily present our framework and the description of the implementation with reference to well known concepts. In a home, we estimated an amount of about 3, 5 TB per year, that are a quite big amount of data. However, the framework can be used to manage more complex buildings, with more evident benefits.

The remainder of the paper is organized as follows. Section 2 describes related works. In Section 3, we motivate our work, and in Section 4 we provide an overview of AllJoyn, highlighting its main features and disadvantages for the development of smart environments in IoT. Our framework for the monitoring and management of smart IoT environment is presented in Section 5. A system prototype for smart home is presented in Section 6. Experiments on a simulated smart home environment and focusing on regular, event, and automated patterns are discussed in Section 7. Section 8 concludes the paper.

2. Related work

Nowadays, Iot devices are deployed in many smart environments [27]. The Internet of Things market is fragmented, amorphous, and continually changing, and several open source software projects, focusing on open source tech for home and industrial automation, are available [33]. IoTivity [11] is an open source software framework enabling seamless device-to-device connectivity to address the emerging needs of the Internet of Things. In October 2016, the AllJoyn and Iotivity project merged, thus devices running either AllJoyn or Iotivity will be interoperable and backward compatible. DeviceHive [12] is developed and commercially supported by a global technology consulting company DataArt. Using DeviceHive, AllJoyn-based device management platform runs on cloud services such as Azure, AWS, Apache Mesos, and OpenStack. DeviceHive focuses on Big Data analytics using tools like ElasticSearch, Apache Spark, Cassandra, and Kafka. OpenIoT [21] is a middleware that aims to facilitate open, large-scale IoT applications using a utility cloud computing delivery model. The platform includes sensor and sensor network middleware, as well as ontologies, semantic models, and annotations for representing IoT objects.

Several middlewares have been proposed for IoT purposes so far. In [8], a novel network system based on an M2M Gateway for smart building is proposed, improving services for users by offering new opportunities for both service providers and network operators. A cognitive management framework for IoT is proposed in [31]. Here, dynamically-changing real-world objects are represented in a virtualized IoT environment, and cognition and proximity are used to select the most relevant IoT objects for the purpose of IoT application in an intelligent and autonomic fashion. Even though such initiatives address the problem of IoT device monitoring and management, they do not consider the big data coming from sensing and actuating devices that require proper storage and analytics solutions. To overcome this limitation, we proposed our new framework. A piece of framework for resource management of Streaming Data Analytics Flows (SDAF) is discussed in [15]. Using advanced techniques in control and optimization theory, an adaptive control system tailored to the data ingestion, analytics, and storage layers of the SDAF is designed. Moreover, it is able to continuously detect and self-adapt to workload changes for meeting the users' service level objectives.

The problem abstract sensing and actuating data was identified in [3], focusing on the concepts of proximity, adjacency or containment. Moreover, a global model with a dynamic interoperability disregarding how the global view should be accomplished was provided. Their decision-maker is able to process incoming big data, but it is not clear how such features are practically implement. In [14], an extension of the Ubiquitous Sensor Networks framework that leverage the Sensor Web Enablement (SWE) standard along with the Session Initiation Protocol (SIP) protocol [13] is proposed. However, one of the main problems of the SIP protocol is related to network constraints. The adoption of the AllJoyn technology, as in our framework, allows us to bypass the problem of uniform communication among heterogeneous devices because AllJoyn provides a common language to interconnect smart devices. Exploiting AllJoyn-based devices allow us to provide a common platform to exchange information independently from the adopted communication system and, hence, develop new applications and services.

The application of the AllJoyn standard in smart environments has been investigated in the literature to offer new solutions for smart environments. In [17], the authors introduce DIO, an android mobile application implemented on the top of the AllJoyn platform that implements efficient communication among co-located people. The app enables people to share and exchange their profiles, favorite photos, musics, videos and documents. Connectivity challenges associated with Smart TVs to ensure a positive user experience together with some solutions that achieve these requirements are presented in [26]. In [25], the authors present ELIoT, that is a development platform for implementing functionality running on networked smart devices, and enabling the integration of such functionality with world wide services through the Internet. The development of AllJoyn based applications has arisen some issues related to AllJoyn. For example, authors in [28] investigated

AllJoyn security aspects, discussing security and privacy issues in deploying AllJoyn compliant devices within homes, buildings and urban infrastructures across the globe. Although the application of the AllJoyn standard in smart environments is widely analyzed in the literature, real time big data oriented services for monitoring smart environments using AllJoyn have not been properly investigated. This motivated us to deal with merging the AllJoyn standard together with storage and computing functionalities. Thus, we provide the current organization of the paper discussing storage/computing platforms together with AllJoyn to provide an effective contribution to the state of the art.

Perspectives and challenges of IoT-based dig data storage systems in Cloud computing have recently been discussed in [6,16,19]. An Integrated IoT Big Data Analytics (IBDA) framework for the storage and analysis of real time data generated from IoT sensors deployed inside the smart building is discussed in [4]. The applicability of the framework is demonstrated with the help of a scenario involving the analysis of real-time smart building data for automatically managing the oxygen level, luminosity and smoke/hazardous gases in different parts of the smart building. An IoT-based system for next generation Super City planning using Big Data analytics is discussed in [24]. In particular, a complete system that includes various types of IoT-based smart systems like smart home, vehicular networking, weather and water system, smart parking, and surveillance objects is proposed for data generation. A Semantic Interoperability Model for Big-data in IoT (SIMB-IoT) to deliver semantic interoperability among heterogeneous IoT devices in health care domain is proposed in [29]. This model is used to recommend medicine with side effects for different symptoms collected from heterogeneous IoT sensors. A collaborative resource management for big IoT data processing in Cloud is proposed in [1]. In order to execute heterogeneous big IoT data handling demands from clients, a Cloud confederation model that determines ideal choices for target cloud providers is discussed. A Dynamic Prime Number Based Security Verification (DPBSV) scheme for big data stream processing scheme based on a common shared key that is updated dynamically by generating synchronized pairs of prime numbers is discussed in [23].

In this paper, storage and retrieval of Big data is a very important task that influence the design of the proposed solution. Indeed, whenever our system need to execute applications with heavy requirements in terms of data storage, some data can be moved at the Edge or on external Cloud resources. Hybrid Cloud-Edge solution to store huge amount of data during the time is not technically discussed in the paper because it is out of the scope of the paper, but we have discussed about this topic in other previously published papers [7,9,30].

3. Motivations

Smart environments are complex systems where several IoT devices are interconnected in order to provide meaningful and reliable information. At the same time, the arising of new technologies for data storage and processing is opening new opportunities in developing new types of services for smart environments, such as energy efficiency management, intrusion detection and improvement of people quality of life. From the analysis of the state of the art, it is evident that several technologies and solutions are available in the literature to address specific issues in smart environments, such as IoT networking, Big Sensing data storage and retrieval, remote Cloud-oriented processing [5]. However, a solution able to integrate such different functionalities in order to implement innovative services is still missing, and, thus, implementation of smart environment prototypes becomes hard. Usually, devices installed in smart environments work alone offering specialized services (such as setting of comfortable room temperature, switching light, and controlling access doors). Moreover, solutions from different producers are not able to interact with each other, thus reducing the opportunity to develop new solution over the same hardware and software components. In our view, all the smart components deployed in an environment should be able to cooperate or, at list, to provide their services to high level software components able to integrate data and services in order to provide additional services. However, vendor specific application architectures and proprietary communication protocols are not the right solution to overcome IoT fragmentation issues.

As example for real application of the proposed solution, we analyzed its applicability in the security management of an Open Campus. A safety-oriented service in an Open Campus has the main requirement to ensure the security of the University campus, both for users, for which security systems need to be less invasive, and for security personnel, for which security systems need to be fast and easy to use. As first goal, intelligent gates to check who can access the Campus (both pedestrian and vehicle gates) need to be installed. These gates will be constantly monitored, for example with cameras (e.g., verification of authorized license plates), because having an up-to-date picture of who is occupying spaces within the University is an important aspect in security management. Security management also needs intelligent access systems that utilize various communication technologies (e.g. NFC, Rfid, SmartPhone, QR-Code, Phonecalls, etc.) for accessing classrooms, laboratories and study rooms. The main issue in addressing this kind of scenario and to implement security oriented services in an Open Campus is the interoperability of heterogeneous systems for intelligent gates and access systems, that have to interoperate for the provisioning of a seamless service. For example, whenever a Professor arrives in the Campus, the platform recognizes him/her and gates for his/her classrooms and laboratories are enabled to be turned on. On the contrary, if a sensor detect fire in a specific area of the campus, the platform enables devices to control if people are in close rooms or areas and, as soon as possible, block access to the burned area.

The solution we aim is open and IoT standards driven, and need to implement the following key features:

- **IoT device networking**: IoT devices interact with the environment and people sensing physical parameters and providing context aware services. They must be interconnected in order to exchange messages and information on related services (e.g., the lighting level according to the number or type of activities of persons in a room), and it follows people movements to always satisfy their needs (e.g., tuning temperature in different rooms or areas of a building according to people current position).
- **Big Data storage**: storage and retrieval of huge amount of data captured from the environment through sensing devices, and produced during data processing and service provisioning. Such data can be stored for long time in order to learn typical behaviors of the environments of people, to detect faults or predict issues.
- **Real time processing**: to provide efficient and responsive services able to improve the Quality of Life of people, information have to be processed in real time, cross relating different types of data, such as audio, video, sensed data, and people preferences.

Our main goal is to develop a framework able to provide all the above features. The framework has to implement distributed technologies, in order to well scale in different application scenarios (e.g., home, building, mall, city, etc.). Also, it is useless to start by scratch, but we can adopt consolidated technologies and solutions that have been adopted in many applications. This approach allows us to:

- reduce development time;
- take part in the scientific communities that develop selected technologies, facing on common issues;
- exploit further functionalities of the adopted technologies that otherwise will not be implemented, in order to provide a more efficient solution.

In the following sections, we will discuss technologies adopted in our framework, their advantages and limits, and how we modified/adapted them for our purposes.

4. IoT applications based on AllJoyn: Main features and limits

Currently, IoT covers multiple application contexts (e.g., personal area networks, home automation, industrial production, smart cities, sensing infrastructures, etc.), and it implies the need to develop new systems with distributed embedded intelligence. AllJoyn is an open source project that provides a universal software framework for the interoperability among heterogeneous devices and for the dynamic creation of proximal networks and distributed IoT applications. It is based on the concept of proximity and mobility providing an environment for the development of ad-hoc, proximity-based, and peer-to-peer (P2P) applications independently from the adopted communication system. Considering that the mobile Peer-to-Peer (P2P) is not easy in Bluetooth networks, an open standard for P2P does not exist, and that the P2P over Cloud is not proximal and it introduces network latency, AllJoyn aims to become the standard for the communication in mobile P2P systems.

Interoperability among heterogeneous devices and for the dynamic creation of distributed IoT applications is the first important issue that it is necessary to overcome to offer new services in smart environments. In our framework, AllJoyn provides a common language to interconnect smart devices thus that things, but also people, processes, and data become part of the ecosystem. Exploiting AllJoyn-based devices allow us to provide a common platform to exchange information independently from the adopted communication system and, hence, develop new applications and services.

In particular, AllJoyn offers the following main mechanisms:

- Discovering proximal devices and applications;
- Ability to adapt itself to specific devices;
- Data transport management between devices through Bluetooth, Wi-Fi, and other technologies;
- Interoperability between different operating systems;
- Efficient and secure data exchange through D-Bus.

The framework enables the development of multiple applications including media sharing, chat rooms, proximal applications, multiplayer games, social applications, applications of home automation, and so on. The physical layer is responsible to manage IP connections using different technologies, e.g., Bluetooth, Wi-Fi, etc. The AllJoyn Routing component, placed on top of the physical layer, is responsible for discovery, connection management, and security. On top of the physical layer, we also find the Standard Libraries that enable the development of two types of client applications: AllJoyn Standard Client (AJST) and AllJoyn Thin Client (AJTC). The AJST applications are adapt to run on general-purpose system (e.g., Microsoft Windows, Linux, OS X, Android, iOS, etc.). Each AJSC application requires a Bus Daemon used for the communication that can be internal or external to the application (bundled). The AJTC is conceived to link through AllJoyn embedded systems by means of micro-controllers installed, e.g., on household electrical appliances, televisions, thermostats, and many other devices. Since these applications are thought for devices with limited hardware capabilities, it is not possible to use Bus Daemons running on devices. In fact, the multi-threading capabilities required by the Bus Daemon might overload the device. Thus, AJTC uses a simple Bus Attachment that enables it to use the Bus Daemon running on other devices: this means that an AJTC application needs to rely on an AJSC to work. Fig. 1 shows an example of AllJoyn system including two hosts, i.e., A and B, and two embedded system, i.e., A and B. Embedded systems A and B join the communication bus by means of two Bus Daemons running on Host B. The features offered by both AllJoyn Routing and Client Libraries are managed by the Service Framework layer including interoperable cross-platform modules providing the following interfaces:

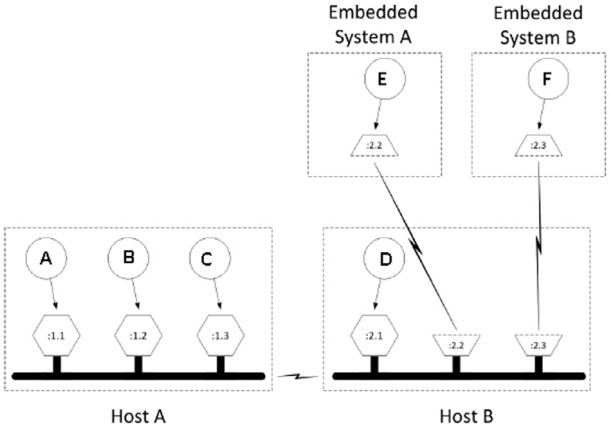

Fig. 1. Example of AllJoyn system including AJSC and AJTC applications.

- Onboarding. Interface that is responsible to discover AllJoyn devices, configure them, and collect data;
- Notifications. Interface that sends/receives human-readable messages configuring priorities, preferences, and that supports the interaction triggered by a notification;
- Control panel. GUI that manages devices' parameters
- Audio. Interface that is responsible to send/receive simple audio messages.

As previously introduced, AllJoyn uses D-Bus for the communication among devices. D-Bus allows us to develop Object-Oriented software independently from the used programming language. In particular, the Bus Daemon of an AJSC applications is responsible for message routing and namespace management. This enables developers to build an ad-hoc bus based on proximal discovery mechanisms. The connection of applications over the D-Bus happens according to a shared namespace. Bus Daemons send advertising messages looking for devices with particular unique well-known names included in the namespace. Once an application, i.e., AJST or AJTC, with a particular name responds to the Bus Daemon, a connection can be established. The connection takes place by means of a session mechanism that allows applications to send/receive events by means of RPC calls. Applications can be developed implementing the interfaces (including methods, signals, and properties) offered by the Service Framework layer.

AllJoyn cannot manage complex large-scale smart IoT environments. In fact, the framework presents several limits in terms of scalability, efficiency, and security. Apart from security, whose implementation is explicitly demanded to application developers, AllJoyn is not able to address two of the main challenges of the IoT: large-scale Smart Environment management and Big Data storage/real-time analytics. AllJoyn does not scale well because it does not support the communication among devices belonging to different broadcast domains (i.e., belonging to different subnets). In fact, it is not possible to think about a worldwide scenario of Cloud Computing including billions of IoT devices connected on

the same broadcast domain. In addition, AllJoyn does not provide any data management system. The billions of devices that will be connected over the Internet by 2020 will introduce on the network a huge amount of data. Thus, it is indispensable to pick out specific data management systems able to process Big Data. To this end, system able to massively process offline Big Data and to process real-time streams of Big Data are strongly required. Both offline and real-time Big Data processing mechanisms are required to develop large-scale smart environments capable to adapt and configure themselves according to events, preferences, and actions performed by users and other "things".

5. A framework for the monitoring and management of smart IoT environments

In this section, we describe how to overcome AllJoyn limitations adopting new technologies for massive data storage and real time processing. The design of our framework start from the assumption that in a smart building there are several heterogeneous IoT devices and embedded and computing systems at the edge of network. Thus, while AllJoyn runs on connected IoT devices and components, other informatics systems (i.e. MongoDB and Storm) act to store and process data.

The introduction of MongoDB in the architecture brings several advantages: (1) it provides a permanent, big data storage of data captured from the environment; (2) it allows us to implement a data-centric communication approach, where AllJoyn devices that are not able to communicate each other can exchange information through the storage system, (3) it allows us to implement an efficient log system to track the activities of the framework. MongoDB replaces the concept of "line" in the relational database to "document" model. Such a storage model is very flexible whenever we do not know in advance the type of data will be monitored. Its unique data storage structure makes it possible to nest documents and the complex hierarchy can be represented by one record. Thus, complex types of data structure can be easily stored. MongoDB has a powerful language query manner similar to object-oriented query, which can satisfy most of the queries in relational database.

As discussed before, the system includes at low level several AJTC applications for controlling different devices, e.g., fans, windows, blinds, thermostats, lamps, and so on. All the AJTC applications belonging to a particular environment are connected to the Bus Daemon running on an environmental AJSC application responsible to manage that particular environment, e.g., a room, a shop, an office, and so on. Furthermore, the environmental AJSC applications can be connected to a centralized AJSC application (e.g., a home, a moll, etc.) that collect data in MongoDB. AJSC application can be deployed in a distributed fashion.

AllJoyn allows real-time device management through P2P connection between devices, a feature present in the design of room AJSCs. In order to preserve the possibility of real-time management, the central system must therefore keep track in real time of the status of the devices and, above all, must be able to manage the events that have occurred in the environment without latencies (e.g. detection of the presence of the smoke sensor, triggering the alarm, or even more banally keep track of the on/off of the devices). To this aim, both the centralized AJSC applications and the various environmental AJSC applications are connected to the Storm system for a global management. Apache Storm is used in the research to analyze spatio-temporal data streams. It has recently emerged as an attractive fault-tolerant open-source distributed data processing platform useful to develop real-time applications for processing a huge amount of data in a scalable manner. With MongoDB and Apache Storm, our system allows us to consider three levels of management: environmental, centralized, and global.

Thanks to Storm and MongoDB, we can manage scalable environments. The Storm cluster includes two types of node: Master and Worker. The Master node runs the Nimbus daemon that is responsible for distributing tasks to the Worker nodes that run a Supervisor daemons. The Coordination between the Master node and Slave nodes is performed by several distributed Zookeeper nodes. This distributed architecture makes the system highly scalable. Storm processes data using the concept of Tuple, i.e., an ordered list of elements. A sequence of Tuples define a Stream. The system provides functionalities to turn a Stream into another Stream by means of Spout and Bolt nodes. Logically, a network of Spouts and Bolts by means of Stream grouping techniques, is defined Topology. In particular, the Spout is a type of node acting as a source of Streams that have to be processed in the Topology. The Bolt is a type of node responsible for the computation that happens in the Topology. It receives data from Spouts and can forward the result to other Bolts for further computations. To collect real-time data coming from AJTC applications, we integrated in each Spout node an AJSC. Thus, for each environmental AJSC, we integrated in the Storm system a particular Topology. Fig. 2 shows an example of logical Storm architecture where Spout nodes collects data coming from different AJSC applications. Considering a scalable scenario, we can have two types of super-peer overlay networks. The first one includes different environmental AJSC applications connected to the Storm system, whereas the second one includes different centralized AJSC connected to the Storm system.

As previously described, each centralized AJSC application is responsible to control several environmental AJSC applications, each one controlling in turn several AJTC applications running on several embedded devices. In addition the centralized AJSC application is responsible to store the collected sensed data in MongoDB for batch data processing. Thus MongoDB will store data in a collection called "device groups". In addition, MongoDB stores data regarding how to control devices in a smart fashion. In fact, data patterns that define the behavior of devices are stored by means of three collections: "regular patterns", "event patterns", and "automated patterns". These collections will be accessed by the Strom system during the real-time data processing of data coming from embedded devices, i.e., AJTC applications. If a sensing device matches a data pattern, a set of actions will be performed by the Storm system. To this end, in each Bolt node, an AJSC will send actions to devices involved in the data pattern. In the following, we describe the three types of data patterns that the architecture supports:

- **Regular patterns** includes a set of actions performed by several embedded devices running AJTC applications that have to be periodically repeated. E.g., from 14 July, 2017 to 18 July, 2017 the coffee machine has to be switched on at 6:45;
- **Event-based patterns** includes a set of actions performed by several devices according to particular events. For example, if sensors detect smoke all windows must be opened;
- **Automated patterns** includes actions not configurable by users, that are triggered according to complex algorithms. E.g., the curtains in a flat can be automatically moved according to the intensity of the external light.

In addition, it is possible to apply a priority to each data pattern.

6. Framework design and implementation

In this section, considering the proposed framework, we describe the design and main implementation highlights of a smart home monitoring and management system. In particular, we will describe how to arrange the basic system and how to develop data patterns for the smart real-time management of embedded devices. In particular, we refer to a smart home case of study for simplicity, in order to provide clear explanation of concepts. However, the proposed framework is scalable, and it can be used to manage

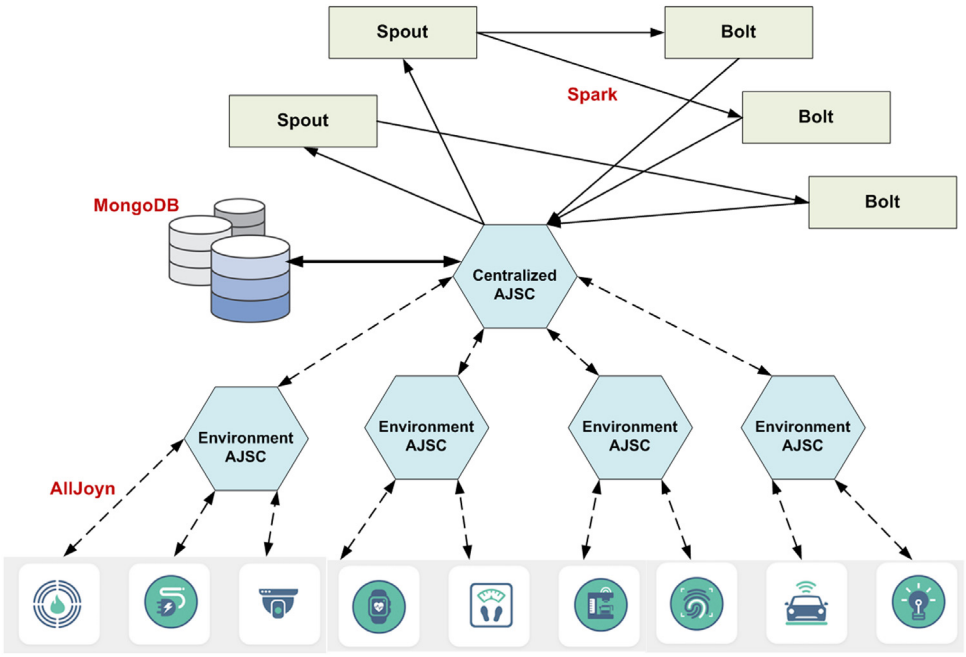

Fig. 2. Architecture of the framework where Spout nodes collect data coming from different AJSC applications.

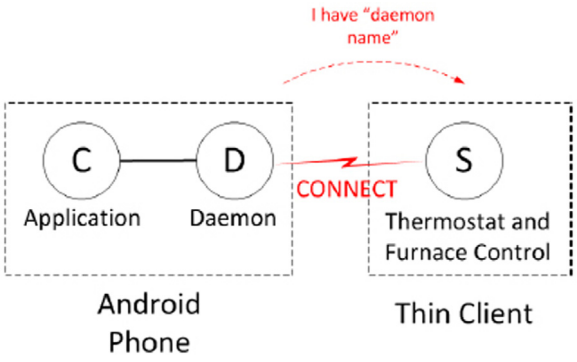

Fig. 3. Example AJTC application running in a thermostat that establishes a connection with a Bus Daemon running in an Android application.

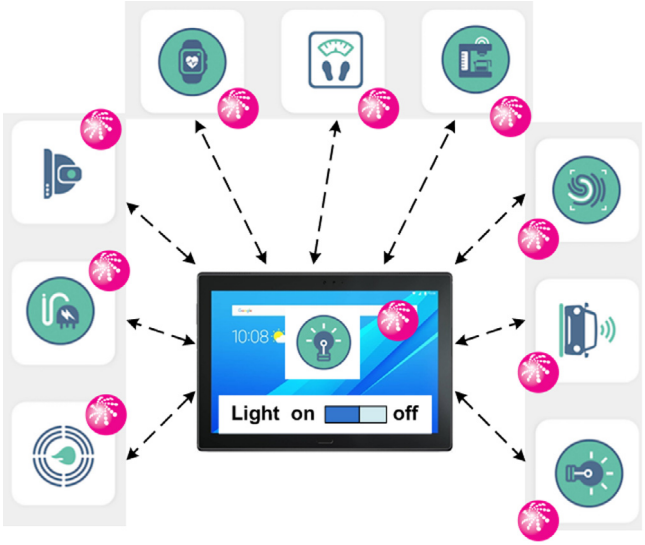

Fig. 4. Example of AJSC application controlling the electrical appliances of a kitchen.

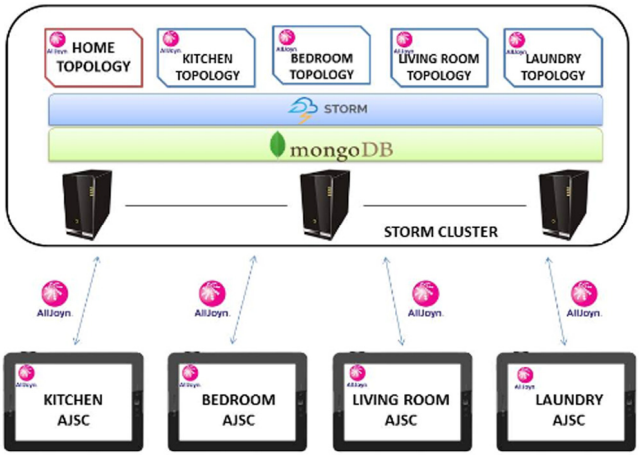

Fig. 5. Example of Smart Home system.

more complex buildings. In particular, its scalability is guaranteed by the scalability of the adopted technologies (i.e., MongoDB and Storm), whereas AllJoyn components can be organized in sub-networks.

6.1. Room AJSC

Let us consider a residence including different smart flats. We want to control and monitor the behavior of devices according to rooms, flats, and the whole residence. The first proximal network that we consider is the room. To this end, we assume that in each room a tablet running Android manages different devices. Android runs an AJSC application and a Bus Daemon used to connect the AJTC applications running on embedded devices. Fig. 3 shows an example of an AJTC running on a thermostat that establishes a connection with the Bus Daemon running in the Android AJSC application. The example in Fig. 3 can be extended to several devices, as shown in Fig. 4, where we show an environmental AJSC application of a kitchen running in a tablet. The AJSC application allows

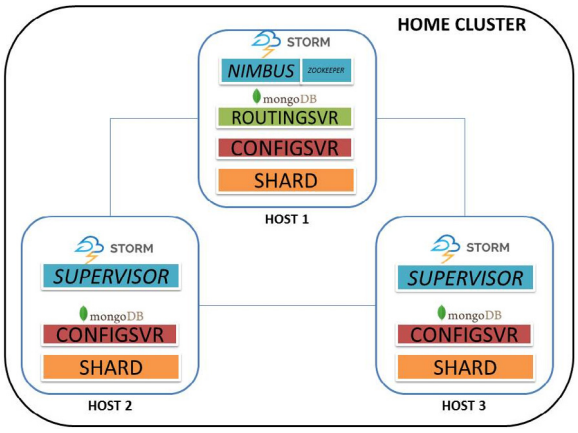

Fig. 6. Example of Home Cluster including three hosts.

us to control the electrical appliances connected to the proximal network of the kitchen. For each smart flat, a new AJSC application allows us to control and manage the AJSC applications running in different rooms. In general, an AJSC application is responsible to facilitate monitoring a portion of a building and to apply data patterns in real-time. In addition, it is responsible to collect sensed data and to store them in MongoDB for further processing.

6.2. Reference architecture

The reference architecture of the whole Smart Home system is shown in Fig. 5. As discussed above, it is possible to control single environments in real-time by using AJSC applications at different rooms/flats, also collecting sensing data. Such data are stored in an instance of MongoDB deployed in a cluster of hosts available in the home. Data stored in MongoDB are processed using a Storm Cluster, according to the logical organization of devices and computers in topologies. To simplify the management of each smart environment, we developed a Storm topology for each room. In addition, we develop a web interface to control the whole system, i.e., for setting up data patterns and for monitoring devices.

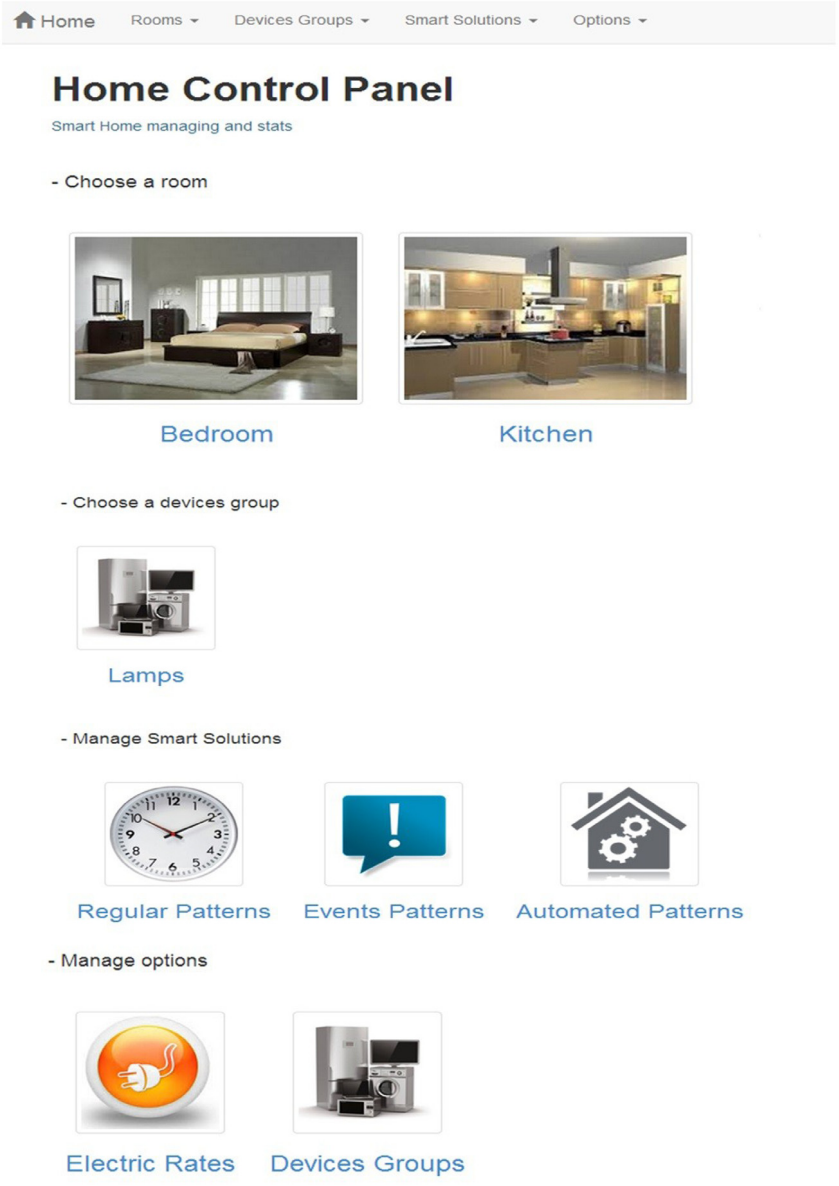

Fig. 7. Example of smart home dashboard.

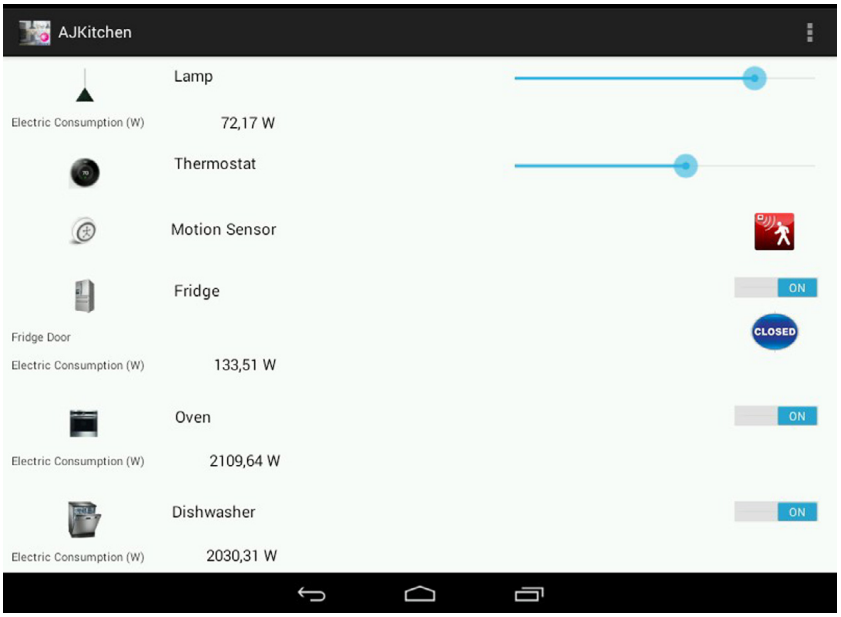

Fig. 8. Example of kitchen dashboard.

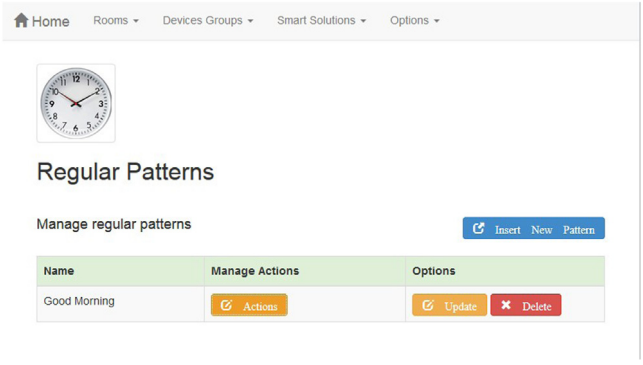

Fig. 9. Example of regular pattern in the kitchen.

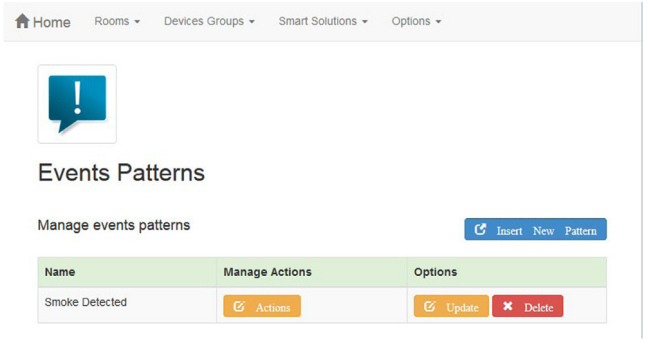

Fig. 10. Example of event-based pattern in the kitchen.

6.3. Home cluster

The storage and processing infrastructure is a distributed Storm/MongoDB system that we call *Home Cluster*. Fig. 6 shows a Home Cluster including three hosts. In each host, we configured Storm and MongoDB. In particular, we configured MongoDB to work with three shards, i.e., replica sets. In addition, we configured one host acting as Storm Master, i.e., running the Nimbus daemon, and two hosts acting as Storm Workers, i.e., running Supervisor daemons. In each, Storm host, we integrated particular AJSC in both Spout and Bolt nodes in order to collect data coming from devices and to control them.

In order to test the system, we emulated the behavior of different electrical appliances placed in a kitchen and we monitored them using the web interface.

6.4. Dashboards

Fig. 8 shows an example of kitchen dashboard. It includes (i) an indicator to control the intensity of lamps and thermostat; (ii) switches to turn on/off fridge, oven and dishwater; and (iii) icons indicating the status of motion and energy consumption sensors. Examples of regular, event-based, and automated patterns in the kitchen dashboard are depicted respectively in Figs. 9, 10, and 11 Fig. 7 shows an example of smart home dashboard. It allows to (i) choose a specific room (e.g., kitchen and bedroom) where to control IoT devices; (ii) choose a service groups (e.g., lamps); (iii) manage smart applications by means of regular, event-based, and automated patterns; and manage options (e.g., electric rates). In our architecture, we refer to Edge devices to implement our framework for several reasons: (1) real time processing requires localized elaboration of data (it is not mandatory, but allows to improve performances); (2) costs of storage systems are decreasing and it is possible to buy TBs at low cost; (3) owner of data could prefer to not put data in eternal clouds for security issue. However, when the amount of data increases, if the system has to execute applications that perform statistics and pattern and predictive analysis over a long period, oldest data can be moved on external Cloud resources. Even if not technically discussed in this paper because out of scope, the framework we developed could also implement an hybrid Cloud-Edge solution to store huge amount of data during the time [7,9,30,32], for example implementing a Local MongoDB database and creating backups into the Cloud using filesystem snapshots, or "block-level" backup methods.

7. Experiments

To test the efficiency of our solution and evaluate its responsivity, we performed some experiments considering a typical smart

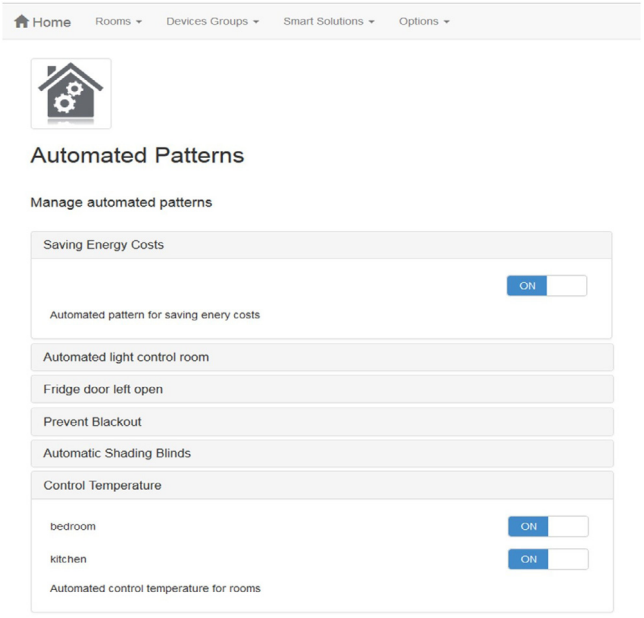

Fig. 11. Example of automated pattern in the kitchen.

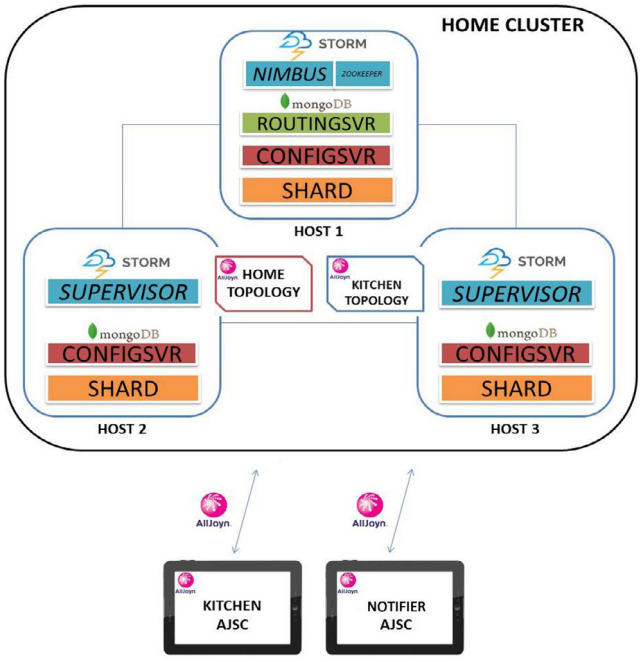

Fig. 12. Scenario implemented in the experiments.

home scenario, with several IoT devices in the kitchen. In particular, we simulated the presence of a coffee machine, a dishwasher, a fan, a fridge and an oven in the kitchen, and we considered sensors for detecting motion and opening of the fridge. Configurations and software components we set up in our scenario are shown in Fig. 12. We deployed one *Room topology*, that is the *Kitchen topology*, to manage smart devices and sensors in the kitchen, and the *Home topology* to manage the whole system. Services and tools are deployed over three hosts that compose the *Home cluster*, as described in Section 6. Devices are governed by different applications, that are the *Kitchen AJSC* and the *Notifier AJSC*. Kitchen AJSC is a Room AJSC specifically designed to manage the Kitchen Topology according to Regular and Automated Patterns. Notifier AJSC is the application responsible to manage notifications of events occurred in the system, when Event Patterns are enabled, in order to trigger new status for devices, or automation in the system.

In order to validate our prototype, we run services and applications for 6 h, implementing triggers and events. In Fig. 13, we report device status triggers in our experimentation, for all the sensing devices exploited. In 6 h, each sensing device performs 360 sensing tasks (one every minute), and each task generated a sensing document in MongoDB of 250 KB. Considering the total amount of devices in the proposed scenario, in 6 h we collect 615 MB of documents. Thus, we can estimate that only the kitchen room could produce about 2,4 GB of documents per day, and a smart home with several rooms could generate about 10 GB per day, and about 3,5 TB per year. This huge amount of data can further increase if we consider a smart building scenario, and clearly requires scalable Big data management solutions to face with data storage and processing issues, as in the architecture we proposed.

In our experiments on a smart home scenario, we evaluated application latency in case of: (1) Regular patterns management, (2) Event-based patterns management, (3) Automated patterns management. Test on the management of Regular Patterns evaluated execution times of Regular Bolts, which performs analysis of data every minute. The average execution time measured is 31 ms and the execution time measured during the time is shown in Fig. 14.

Differently from Regular Patterns, Event Patterns are managed by the Home Topology as response to a particular event. Whenever an event occurs, the Bolt of Event Patterns checks if specific actions have to be performed by devices, and sends the notification. In Fig. 15, the time spent to manage events during the time is drawn. By increasing the time, the number of running patterns increases and the average time spent to complete a task at the Bolt decreases till 72 ms.

Automated Patterns can be of two types: (1) Room-based or (2) Home-based. Room-based Automated Patterns define policies related to a specific Room (e.g., tuning room temperature, tuning lightness according to people in the room or the time, etc.), whereas Home-based Automated Patterns involve the whole home or building (e.g., management of blackout, energy saving strategy, etc.). Algorithms executed to process Automated Patterns can be complex and performance is related to the complexity of such algorithms. This implies that usually execution time of Bolts for Automated Patterns is higher than Regular and Event Patterns. However, the latency of the service strongly depends on the type of application. For example, in our experiments, the average time spent by the Bolt Room to manage room temperature is about 20 ms, as shown in Fig. 16. The average time spent by the Bolt Room to manage lightness is higher, and about 110 ms (see Fig. 17). Fig. 18 shows the average latency for the Home Bolt to perform energy saving. We measured a latency of 97 ms on average, and it proves that latency for Automated Patterns does not depend on the pattern type (e.g., Room or Home-based), but on the specific service it implements.

8. Conclusion

In this paper, we focused on the monitoring and management of IoT devices in a smart environment producing big sensing data. In particular we discussed the limitation of AllJoyn and we proposed a framework integrating it with MongoDB and Storm. In addition, we discussed how the system allows us to overcome the limitation of AllJoyn in terms of large-scale Smart Environment monitoring/management and big data storage/analytics. Finally, a smart

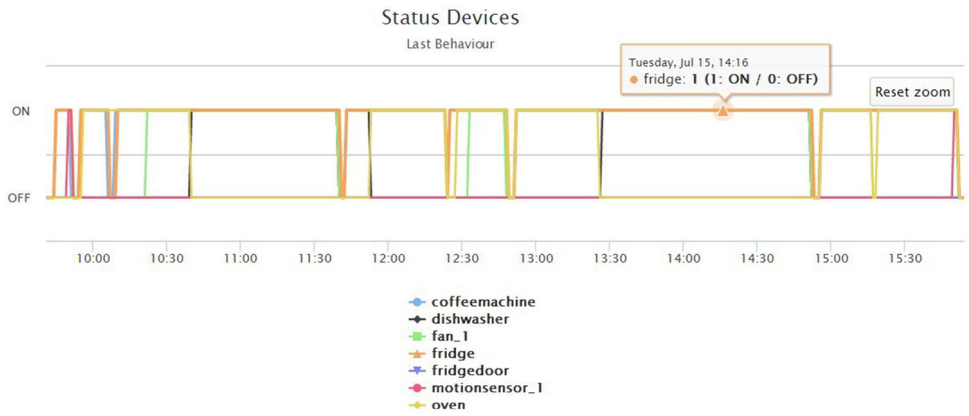

Fig. 13. Example of device sensing triggers.

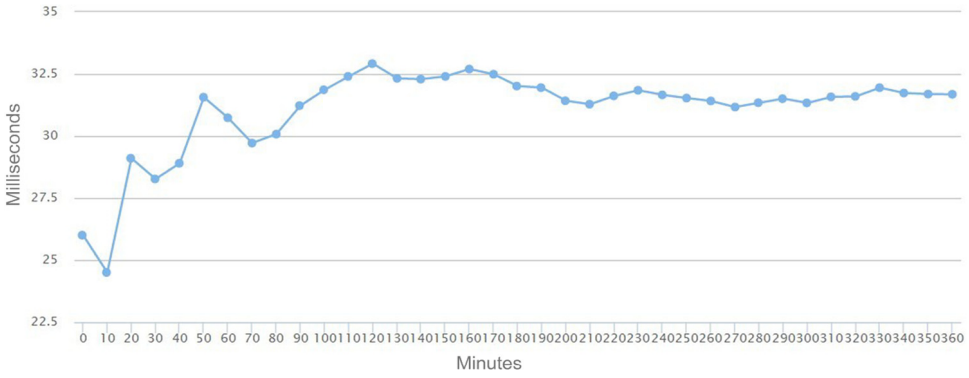

Fig. 14. Measurements with Regular Pattern management.

Fig. 15. Measurements with Event Pattern management.

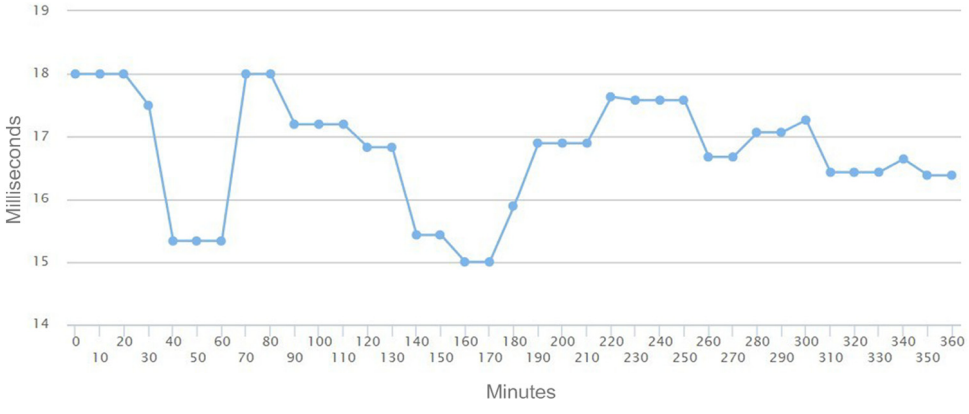

Fig. 16. Measurements with Automated Pattern management for room temperature tuning.

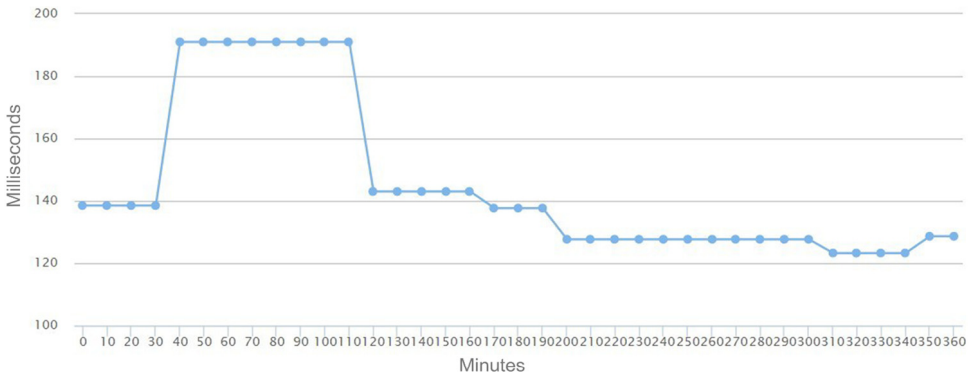

Fig. 17. Measurements with Automated Pattern management for lightness tuning.

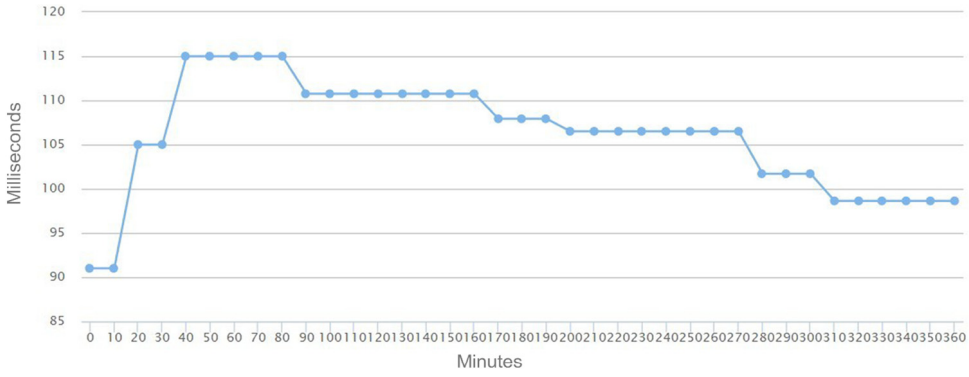

Fig. 18. Measurements with Automated Pattern management for energy saving.

home case of study was analyzed, specifically focusing on regular, event-based and automated patterns aimed at controlling IoT devices in a smart fashion. Experiments conducted in a simulated kitchen prove the goodness of our system in terms of response time. Our solution can be easily adopted in largest smart environment such as smart mall and smart cities. Currently, our piece of framework requires the manual configuration of each plugged IoT device. In future works, we plan to extend our framework in order to introduce autonomic secure IoT configuration mechanisms.

References

[1] A. Alelaiwi, A collaborative resource management for big iot data processing in cloud, Cluster Comput. 20 (2) (2017) 1791–1799.
[2] Apache, Storm, http://storm.apache.org.
[3] D. Ballari, M. Wachowicz, M.A. Manso, Metadata behind the interoperability of wireless sensor network, Sensors 9 (2009) 3635–3651.
[4] M. Bashir, A. Gill, Towards an iot big data analytics framework: Smart buildings systems, in: Proceedings - 18th IEEE International Conference on High Performance Computing and Communications, 14th IEEE International Conference on Smart City and 2nd IEEE International Conference on Data Science and Systems, HPCC/SmartCity/DSS 2016, 2017, pp. 1325–1332.
[5] A. Bujari, M. Ciman, O. Gaggi, G. Marfia, C.E. Palazzi, Paths: Enhancing geographical maps with environmental sensed data, in: Proceedings of the 2015 Workshop on Pervasive Wireless Healthcare, in: MobileHealth '15, ACM, New York, NY, USA, 2015, pp. 13–16.
[6] H. Cai, B. Xu, L. Jiang, A. Vasilakos, Iot-based big data storage systems in cloud computing: Perspectives and challenges, IEEE Internet Things J. 4 (1) (2017) 75–87.
[7] A. Celesti, M. Fazio, A. Galletta, L. Carnevale, J. Wan, M. Villari, An approach for the secure management of hybrid cloudedge environments, Future Gener. Comput. Syst. 90 (2019) 1–19.
[8] R. Fantacci, T. Pecorella, R. Viti, C. Carlini, A network architecture solution for efficient iot wsn backhauling: challenges and opportunities, IEEE Wirel. Commun. 21 (4) (2014) 113–119, http://dx.doi.org/10.1109/MWC.2014.6882303.
[9] M. Fazio, A. Celesti, A. Puliafito, M. Villari, Big data storage in the cloud for smart environment monitoring, Procedia Comput. Sci. 52 (2015) 500–506, the 6th International Conference on Ambient Systems, Networks and Technologies (ANT-2015), the 5th International Conference on Sustainable Energy Information Technology (SEIT-2015).
[10] O.C. Fundation, AllJoyn, https://openconnectivity.org/developer/reference-implementation/alljoyn.
[11] L. Fundation, IoTivity, https://www.iotivity.org/.
[12] L. Fundation, DeviceHive, https://devicehive.com/.
[13] J.M. Hernández-Muñoz, J.B. Vercher, L. Muñoz, J.A. Galache, Presser, Ubiquitous sensor networks in ims: an ambient intelligence telco platform, in: ICT Mobile Summit, Stockholm, 2008.
[14] J.M. Hernández-Muñoz, J.B. Vercher, L. Muñoz, J.A. Galache, M. Presser, L.A.H. Gómez, J. Pettersson, The future internet, in: J. Domingue, A. Galis, A. Gavras, T. Zahariadis, D. Lambert (Eds.), Lecture Notes in Computer Science, Springer-Verlag, Berlin, Heidelberg, 2011, pp. 447–462, Ch. Smart cities at the forefront of the future internet. URL http://dl.acm.org/citation.cfm?id=1983741.1983773.
[15] A. Khoshkbarforoushha, A. Khosravian, R. Ranjan, Elasticity management of streaming data analytics flows on clouds, J. Comput. System Sci. 89 (2017) 24–40.
[16] A. Khoshkbarforoushha, R. Ranjan, R. Gaire, E. Abbasnejad, L. Wang, A.Y. Zomaya, Distribution based workload modelling of continuous queries in clouds, IEEE Trans. Emerg. Top. Comput. 5 (1) (2017) 120–133.
[17] J. Kwak, J.-H. Jin, M.-J. Lee, A mobile application for information sharing and collaboration among co-located people, Inf. Technol. Comput. Sci. 106 (2015) (2015) 17–21.
[18] Z. Liang, Q. Zhang, A new method of controlling iot devices based on cloud storage service, in: 2017 IEEE International Conference on AI Mobile Services (AIMS), Honolulu, Hawaii, United States, 2017, pp. 113–116.
[19] C. Liu, R. Ranjan, X. Zhang, C. Yang, D. Georgakopoulos, J. Chen, Public auditing for big data storage in cloud computing – a survey, in: 2013 IEEE 16th International Conference on Computational Science and Engineering, 2013, pp. 1128–1135, http://dx.doi.org/10.1109/CSE.2013.164.
[20] MongoDB, https://www.mongodb.com.
[21] OpenIoT, http://www.openiot.eu/.
[22] C.E. Palazzi, S. Ferretti, M. Roccetti, G. Pau, M. Gerla, What's in that magic box? the home entertainment center's special protocol potion, revealed, IEEE Trans. Consum. Electron. 52 (4) (2006) 1280–1288.
[23] D. Puthal, S. Nepal, R. Ranjan, J. Chen, Dpbsv – an efficient and secure scheme for big sensing data stream, in: 2015 IEEE Trustcom/BigDataSE/ISPA, Vol. 1, 2015, pp. 246–253, http://dx.doi.org/10.1109/Trustcom.2015.381.

[24] M. Rathore, A. Paul, A. Ahmad, G. Jeon, Iot-based big data: From smart city towards next generation super city planning, Int. J. Semant. Web Inf. Syst. 13 (1) (2017) 28–47.
[25] A. Sivieri, L. Mottola, G. Cugola, Building internet of things software with eliot, Comput. Commun. 89–90 (2016) 141–153, http://dx.doi.org/10.1016/j.comcom.2016.02.004, internet of Things Research challenges and Solutions URL http://www.sciencedirect.com/science/article/pii/S0140366416300238.
[26] M. Stauffer, Connectivity solutions for smart tvs, in: 2012 IEEE Second International Conference on Consumer Electronics - Berlin (ICCE-Berlin), 2012, pp. 245–249.
[27] R. Tabish, A. Ghaleb, R. Hussein, F. Touati, A. Ben Mnaouer, L. Khriji, M. Rasid, A 3g/wifi-enabled 6lowpan-based u-healthcare system for ubiquitous real-time monitoring and data logging, in: Biomedical Engineering (MECBME), 2014 Middle East Conference on, 2014, pp. 277–280, http://dx.doi.org/10.1109/MECBME.2014.6783258.
[28] O. Tomanek, L. Kencl, Security and privacy of using alljoyn iot framework at home and beyond, in: 2016 2nd International Conference on Intelligent Green Building and Smart Grid (IGBSG), 2016, pp. 1–6.
[29] F. Ullah, M. Habib, M. Farhan, S. Khalid, M. Durrani, S. Jabbar, Semantic interoperability for big-data in heterogeneous IoT infrastructure for healthcare, Sustainable Cities Soc. 34 (2017) 90–96.
[30] M. Villari, M. Fazio, S. Dustdar, O. Rana, R. Ranjan, Osmotic computing: A new paradigm for edge/cloud integration, IEEE Cloud Comput. 3 (6) (2016) 76–83.
[31] P. Vlacheas, R. Giaffreda, V. Stavroulaki, D. Kelaidonis, V. Foteinos, G. Poulios, P. Demestichas, A. Somov, A. Biswas, K. Moessner, Enabling smart cities through a cognitive management framework for the internet of things, IEEE Commun. Mag. 51 (6) (2013) 102–111.
[32] D. Weerasiri, M.C. Barukh, B. Benatallah, Q.Z. Sheng, R. Ranjan, A taxonomy and survey of cloud resource orchestration techniques, ACM Comput. Surv. 50 (2) (2017) 26:1–26:41.
[33] C.W. Wu, F.J. Lin, C.H. Wang, N. Chang, Onem2m-based iot protocol integration, in: 2017 IEEE Conference on Standards for Communications and Networking (CSCN), 2017, pp. 252–257.

Dr. Antonio Celesti received the Ph.D. in "Advanced Technology for Information Engineering" in 2012 at the University of Messina (Italy). From 2008, he is one of the members of the Mobile and Distributed Systems Laboratory (MDSLab) in Messina. He worked as collaborator in several International projects including EU FP7 RESERVOIR, EU FP7 VISION CLOUD, EU FP7 frontierCities and EU Horizon 2020 BEACON.

From December 2015, he is a technical–scientific fellow at the Scientific Research Organizational Unit, University of Messina.

He is currently Adjunct Professor of Electronic and Computer Science Bioengineering and database II. His main research interests include distributed systems, Cloud computing and IoT service federation and security.

Maria Fazio has been involved in many national and international projects, including the EU FP7 CloudWave Project (2013–2016) and EU Horizon 2020 BEACON (2015–17). She is Co-Editor-In-Chief of the EAI Endorsed Transactions on Cloud Systems, and member of the Editorial Board of international journals (e.g., the International Journal of Business Data Communications and Networking (IGI Publishing) and the EAI Endorsed Transactions on Smart Cities). She has been also editor of Special Issues in international journals (e.g., IGI Global-IJDST, IGI Global-IJBDCN, Scalable Computing: Practice and Experience). She acted as chair and organizer in international conferences and workshops (e.g., IEEE-ISCC, EAI-CN4IoT, IFIP-ESOCC, etc.). Her main research interests include distributed systems and wireless communications, especially with regard to the design and development of Cloud solutions for IoT services and applications.