A Cloud-Based Platform of the Social Internet of Things

Roberto Girau, Salvatore Martis, and Luigi Atzori(✉)

Department of Electrical and Electronic Engineering,
University of Cagliari, 09123 Cagliari, Italy
{roberto.girau,salvatore.martis,l.atzori}@diee.unica.it

Abstract. The huge numbers of objects connected to the Internet and that permeate the environment we live in are expected to grow considerably, causing the production of an enormous amount of data to be stored, processed and made available in a continuous, efficient, and easily interpretable manner. Cloud computing can provide the virtual infrastructure that meets these requirements providing the appropriate flexible and powerful tools.

This paper presents a platform that goes in this direction relying on the following features: the PaaS (Platform as a Service) model is fully exploited, for an easy management and development of applications by both users and programmers; each object is an autonomous social agent running in the cloud, according to which objects are capable of establishing social relationships in an autonomous way with respect to their owners with the benefits of improving the network scalability and information/service discovery; the data is under the control of the users, as the data generated by the objects is stored in the objects owners cloud spaces. The paper concludes presenting the implementation of the platform in the Google App Engine PaaS.

Keywords: Social network · Internet of Things · Cloud computing · IoT platform · IoT architecture

1 Introduction

Society is moving towards an always connected paradigm, where the Internet user is shifting from persons to things, leading to the so called Internet of Things (IoT) scenario. In this respect, successful solutions are expected to embody a huge number of smart objects identified by unique addressing schemes providing services to end-users through standard communication protocols. Accordingly, the huge numbers of objects connected to the Internet and that permeate the environment we live in is expected to grow considerably, causing the production of an enormous amount of data that must be stored, processed and made available in a continuous, efficient, and easily interpretable manner. Cloud computing can provide the virtual infrastructure that meets these requirements and can integrate

B. Mandler et al. (Eds.) IoT 360° 2015, Part I, LNICST 169, pp. 77–88, 2016.
DOI: 10.1007/978-3-319-47063-4_7

sensors, data storage devices, analytic tools and artificial intelligence, management platforms providing services to end-users. Additionally, the pricing model on consumption of cloud computing, enables end-to-end services and access to on demand applications and in any place. At the same time, service oriented technologies, web services, ontologies, semantic web also allow for constructing virtual environments for industrial production and services [1]. Indeed, virtualization technologies can hide the physical characteristics of industrial equipment and devices in general implementing an effective connection, communication and control between the real world and the virtual counterpart.

In the last five years many IoT architectural proposals and implementations appeared in the literature and in the market. A great effort has been devoted in defining architectures and relevant layers functionalities around the concept of virtualizing the physical objects. This is exploited to improve resilience, service discovery and composition as well as to enhance ubiquity. Some of the implementations have been also designed to exploit the cloud computing features, often for the realization of vertical solutions addressing specific application domain requirements. As it is discussed in the following section, we still believe that to fully exploit the potentialities of the IoT paradigm, there is a need for further advancements in designing platforms that: make even easier the communications among objects; help the work of the developers in creating new applications on top of the available objects services; allow the users to have complete control of their own data and objects; are reliable and efficient to support the interaction of trillions of objects.

To further advance in this respect, this paper presents Lysis[1] a cloud-based platform that exhibits the following features: the PaaS (Platform as a Service) model is fully exploited, for an easy management and development of applications by both users and programmers; each object is an autonomous social agent running in the cloud, according to which objects are capable of establishing social relationships in an autonomous way with respect to their owners with the benefits of improving the network scalability and information/service discovery; the data is under the control of the users, as the data generated by the objects is stored in the objects owners cloud spaces. The paper also presents the implementation of the Lysis in the Google App Engine PaaS.

The paper is organized as follows. Section 2 provides some background information about the past architectural solutions and the use of cloud computing and virtualization solutions. Section 3 describes the major layers of the Lysis platform and the key functionalities. Sections 4 and 5 present some implementation details and final conclusions, respectively.

2 Background

The intention of this section is to briefly review the major works in the three major areas of interest for this work.

[1] Lysis is the only dialogue of Plato in which the philosopher Socrates discusses the nature of friendship with his disciples.

2.1 IoT Architectures and Objects Virtualization

The main objective of the iCore project [2,3] is to provide a framework for (almost) autonomous IoT application development and it refers to a three level architecture: in the virtual object (VO) level, virtual alter ego of any real-world object (RWO) are dynamically created and destroyed; cognitive technologies guarantee a constant link between RWO and VO and ensure self-management and self-configuration. The iCore framework, has a virtualization level which provides the following functionalities: creation process, naming, addressing, discovery, security, privacy, interfacing and communication. These functionalities are exploited by dedicated central elements like registry, repository and management servers. In the above layer, VOs are aggregated in CVOs (Composite Virtual Objects) to meet application requirements. The last layer is the service layer, which has the role to translate the application requirements into services to be fulfilled by the CVO level through the exploitation of artificial intelligence systems. The iCore team has developed a preliminary prototype, which however has not been devised for being deployed in the cloud. In a similar way, other projects such as COMPOSE [4] and IoT-A [5] exploit virtual counterparts(Service Objects and Virtual Entities respectively) to create IoT applications. The FI-WARE platform [6,7] is based on elements (Generic Enablers) that permit re-usability and allow for sharing functions on a multiplicity of areas of use in the Internet. In the specific field of the IoT, the FI-WARE has mostly proposed GEs for handling objects communications, resource management, process automation. These are made available to be integrated in other platform but there is not a common framework for an easy deployment of these GEs.

2.2 Cloud in IoT

Many platforms exploit cloud computing features to provide IoT services in industrial environment such as smart home [8], smart cities [9], smart management of inventories [10] and production [11], iHealth [12], environmental monitoring [13], social security and surveillance [14], mine security [15], Internet of Vehicles (IoV)[16]. Although very effective for the purpose they have been proposed, these solutions are too vertical, lacking in horizontal enlarge-ability(cross-applications), de facto limiting their adoption in other IoT application domains. Indeed, in these realizations, domain-specific or project-specific requirements drove the design of all the system components and determine most technological elements ranging from sensors and smart devices to middleware components and application logic. Consequently, most of IoT applications in cloud are designed with a vertical approach as discussed in [17]. In this paper the authors highlight that isolated IoT platforms are implemented like silos and have been also named virtual verticals. Accordingly, any client of IoT solutions owns her own solution virtually isolated from the others and just share the storage and computing resources. They then propose an additional component, named domain mediator to make the different PaaS IoT platforms talk each-other. This issue is also the focus of Gubbi et al. that present a user-centric cloud based model to design new

IoT applications through the interaction of private and public cloud showing an attempt of using cloud computing to provide horizontal solutions [18].

2.3 Distributed Social Objects

There are recent studies demonstrating that the issues related to the management and effective exploitation of the huge numbers of heterogeneous devices could find a solution in the use of social networking concepts and technologies. In [19] the authors introduced the idea of objects able to participate in conversations. In [20] things are involved in social networks with humans. In [21,22], explicitly, the Social IoT (SIoT) concept is formalized, which is intended as a social network where every node is an object capable of establishing social relationships with other things in an autonomous way according to rules set by the owner. Following the specific SIoT paradigm presented in [22], an IoT platform from the open source project ThingSpeak [23] has been developed, which has been extended augmenting the objects with the social attitude. Accordingly, the objects can create the following relationships: the *Ownership Object Relationship* (OOR) is created between objects that belong to the same owner; the *Co-location Object Relationship* (CLOR) is created between stationary devices in the same place, as can be appliances of a dwelling; the *Parental Object Relationship* (POR) is created between objects of the same model, producer and production batch; the *Co-work Object Relationship* (CWOR) is created between objects that meet each others in the owners' workplace as the laptop and printer in the office; the *Social Object Relationship* (SOR) is created as a consequence of frequent encountering between objects, as can happen between smartphones of people who use the same bus every day to go to school/work, people hanging out at the same bar/restaurant/gym.

These features have however been implemented making use of a centralized approach that does not exploit the benefits of a distributed approach that can be achieved by allowing object-to-object direct and autonomous communications.

3 The overall Lysis architecture

In this paper we present an IoT platform called Lysis, which presents four basic features:

- Distributed social objects. The integration of social networking concepts into the Internet of Things has led to the Social Internet of Things paradigm, according to which objects are capable of establishing social relationships in an autonomous way with respect to their owners with the benefits of improving the network scalability in information/service discovery.
- Virtualization and PaaS model. Virtual objects implement the digital counterparts of the physical devices, speak for it and introduce some functionalities that could not be taken by the real world objects such as: supporting the discovery and mash up of services, fostering the creation of complex applications, improving the objects energy management efficiency, as well as making

the inter-objects communications possible by translating the used dissimilar languages. The cloud is the best environment where computing and storage resources can be assigned to the virtual object in a flexible way.

– Data ownership. Users are granted with the required cloud space for storing sensor data and to run simple applications such as trigger actuations, send alerts and visualize log graphs. In the near future, these services will permeate our everyday activities with all our devices connected in the cloud. This process is felt as a strong threat to the user privacy. There is the need for solutions that assure to the user the complete control of the data, which should then be stored in her reserved space.
– Re-usability. Requests for the same data from the same sensors from different IoT applications cause extreme inefficiency in accessing hardware and result in waste in terms of energy consumption and bandwidth, if not handled properly. Re-usability also refers to the code. Specifically, instantiating a new process for the communication with the physical device and processing of the data should not require rewriting code already developed but should relied on sharing of codes among the communities of users and developers. The sharing should be done at all the architectural layers, from the physical devices drivers till the upper most application layer.

Figure 1 shows the overall architecture of the Lysis platform through four functional levels: the lower level is made up of the "things" in the real world; the one above is the virtualization level, which interfaces directly with the real world and is made up of Social Virtual Objects (SVOs); the level of aggregation is responsible for composing different SVOs to set up entities with augmented functionality called Micro Engines (ME); the last level is the application level in which user-oriented macro services (APP) are deployed. In the following subsections we describe the major components, with particular attention to the major above mentioned features.

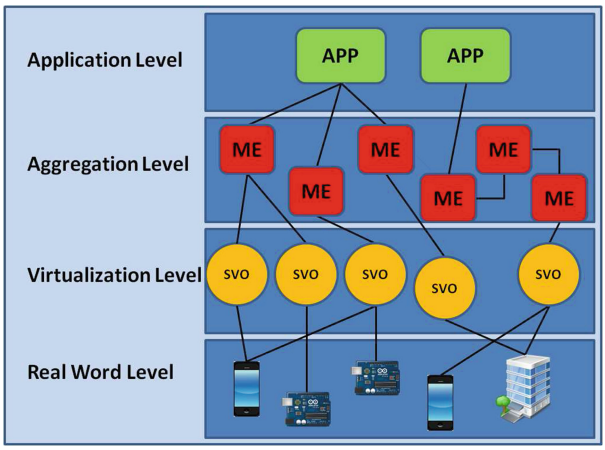

Fig. 1. The four levels of the Lysis architecture.

3.1 The Real World Level

As well-investigated in the iCore project [3], the lowest level is always made up of the Real-World Objects (RWO). Some of these are Physical Devices (PD) capable to directly communicate through the Internet, such as smartphones, laptops and TV set-top-boxes (see Fig. 2(a)). Some others cannot directly access to the internet and have to use local gateways (GW) (see Fig. 2(b)). The PDs and the GWs implement the following modules to be part of the platform:

- Hardware Abstraction Layer (HAL): it communicates with the corresponding module in the virtualization level. Its major role is to introduce a standardized communication procedure between the platform and the extremely variegate set of PDs, simplifying the platform southbound APIs. It is also in charge of creating a secure point-to-point communication (encrypted) with the SVOs.
- Data Handler: it may intervene when there is the need to process data from sensors before being sent by the PD-HAL to the virtualization level. For example, data coming from sensors could be strings of hexadecimals, which have to be processed to extract actual numerical values to encapsulate them in JSON format ready for dispatching.
- Device Management: it implements the real device logic with reference to the participation of the PD to the Lysis platform. It implements most of times simple but key operations like controlling the sensing frequency, managing local triggers, overseeing the energy consumption. It also allows for the running of code that can be updated in run-time locally in the PD.
- Environment interface / protocol adapter: in the case of the PD, it consists in the hardware drivers for all local sensors and actuators. In the case of the GW, it implements the communication with the ICT objects through the available protocol.

3.2 The Virtualization Level

The hardest challenge of the IoT is to be able to address the deployment of applications involving heterogeneous objects, often moving in large and complex environments, in a way that satisfies the quality requirements of the application itself, while not overloading the network resources. For this reasons, the virtual object has become a key component of many IoT platforms, representing the digital counterpart of any real (human or lifeless, static or mobile, solid or intangible) entity in the IoT. It supports the discovery and mash up of services, fostering the creation of complex applications, improving the objects energy management efficiency, as well as addressing heterogeneity and scalability issues.

In our implementation this entity has also a social behavior and for this reason it is named Social Virtual Object. Indeed, it is the abstraction of the RWO in terms of functionalities (i.e., the VO) with a social capability extension (Social Enabler), as shown in Fig. 3. The "Type" of RWO is represented by a *Template* of VO. For example, every smartphone model has the same VO template; however, there is a different VO instance per smartphone PD, which

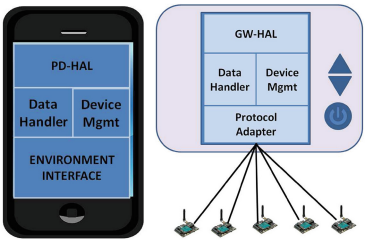

Fig. 2. (a) Physical Devices (PDs) capable to communicate with the platform and (b) objects that need a gateway to interact with the platform.

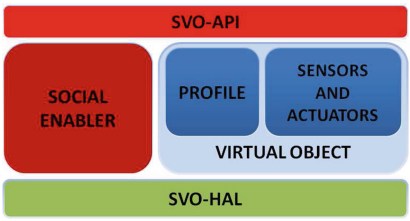

Fig. 3. The components of the Social Virtual Object

is the actual web service running in the cloud. The template consists of the VO Schema, a semantic description of the related RWO. Capabilities and resources of the real object are depicted inside the VO Schema. The second component of the template is the *Software Agent* source code, which is the computational engine of the VO to be run in the cloud.

The VO Schema can be seen as the semantic description of the class of RWOs of the same type, while the VO Profile is a precise description of the object itself. It is important for the installer to complete the semantic description of the instance of VO to allow a correct search of the resources needed for the creation of services. Data points in the VO represent sensors and actuators and are available through REST APIs, usable by the levels above.

The Social Enabler (SE) extends the functionalities of the VO and, consequently, the related Real World Object by adding social capabilities. The SE is in charge of the socialization of the SVO by allowing the establishment, management and termination of social relations. A social graph connecting each SVO with the others according to their friendships is used to find the services required at the application level. Through type and strength of relationships a trustworthiness value of an object is evaluated to provide a desired service [24].

At SVO level, three classes of permissions are foreseen: public, private and friend. In the first case accessing to the resources is allowed to anyone without the need of any API Key. If the permissions are set to "private" the Owner Key is required. Of course, in this case, applications instantiated by the Owner only are allowed to access to resources. Lastly, if the permissions are set to "friend", the access is allowed only by SVO friends which have a friend API Key.

The SVO search is the functionality the application layer is provided with when there is the need of a service and/or information that can be provided by other objects. A key role is taken by a node called SVO Root (SVOR), which is elected among all the SVOs owned by the same owner. The SVOR accepts requests from the upper levels. Once the SVOR is activated, the first action is to check if the required profile matches its own profile. In this case, the SVOR

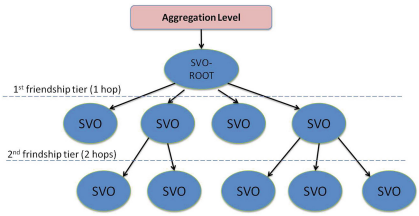

Fig. 4. Objects involved in a SVO search process

responds with its own resources. If it is not the case, it checks its local database if there are matches among its friends. In case of positive result, the SVOR returns the address of the found resource(s) (more than one node may match the profile) and the friend API Key(s) to access to it. In the case of mismatch, the query is forwarded to its friends with high potentials to know the target node or with links with strong network hubs, as shown in Fig. 4. The process is repeated until a positive result is found, which is then returned to the SVOR that sends it to the higher levels. The SVO Root is elected among the SVO in OOR (then belonging to the same user) and among these, the one with the highest value of centrality in the social graph. Since each SVO is able to respond to SVO Search queries, other SVOs in OOR provide the necessary redundancy to the SVOR in case of congestion or malfunctioning. The strength of this system is that there are no single points of failure, and in the case of failing nodes, the network adapts itself by forwarding the requests towards alternative routes of the social graph. In addition, using the SVO with greater centrality decreases the chance of forwarding the request outside the SVO Root.

To deploy SVOs in the cloud, Lysis provides the infrastructural elements shown in Fig. 5. From the Template repository the user chooses the correct Template for the installation of each SVO. The Template is then taken as input by the SVO Deployer, which is in charge of instantiating the agent and giving an initial configuration to SVO. The Deployer works only during the set up phase because once instantiated the SVO is an autonomous web service able to introduce self-updates and manage the communications with its friends as in a human Social Networks.

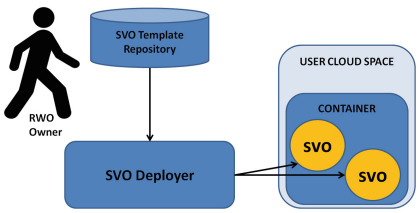

Fig. 5. The SVO deployment process

3.3 The Aggregation Level

The Micro Engine is an entity that is created to implement part of the applications running in the upper layer. It is a mash-up of one or more SVOs and other MEs. With reference to this entity there are two important components: the *Instance* and the *Schema*. The Instance is a piece of programming code running in the cloud. It must be able to reuse the output of an instance to respond to requests that present the same inputs in order to save redundant data requests that consume bandwidth and CPU unnecessarily. It must also be able to understand whether there is a malfunction of one of the input or output, and in this case, requires the reassignment of resources to the control unit. Each ME is described by a Schema that contains a semantic description of the input, the output (if any) and the activity of processing. It also contains a summary of the help that is useful as support for developers using MEs with applications.

Figure 6 shows the elements of the aggregation level. Herein, SVO resources are combined in different MEs, which are entities that inherit some or all of the functionalities of the SVOs and are augmented with more advanced features such as: statistical analysis, data forecasting, artificial intelligent cooperation etc. Associations between MEs and SVOs are managed by the ME Controller. During this phase, the controller triggers the execution of the search operation to find the right SVO and retrieve the related permissions. This SVO Search functionality is the one implemented by the root SVO of the user where the App is running. To be found by the ME Controller, each ME has to be documented in the registry. This element of the aggregation level contains a database of active MEs. Each line of the DB is related to a single ME and contains: the ME ID, the ME URLs, the access permissions, and the time-stamp of the last check.

The Micro Engine Controller is the coordination element of the entire level. When an application at the upper level sends a query for the first time, the MEC checks all the involved ME asking for the related URLs to ME Registry. It asks for SVO Search to the SVO Root of the user who started the application from which takes the owner key. Once it has the resources by the SVOs, it associates

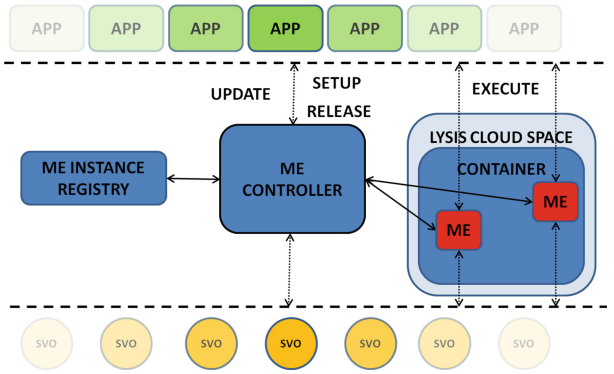

Fig. 6. The ME deployment process

them to the MEs which register the query ID and the required resources (input and output) in the local database. Finally, the MEC notifies the latest ME in the processing chain to the application. That ME will also be the one elected as responder which responds to the queries of the application. communication with its friends as in a human Social Network. Once instantiated, the SVO runs in the user cloud space.

3.4 The Application Level

At this level the applications are deployed and executed exploiting one or more Micro Engines. The interface with the user also assumes a key role; in fact, although we are in the field of the IoT solutions, which are centered around device-device communications, the center of gravity at the end is still the user. An application at this level shows a font-end interface to the user, and a back-end interface to the underlying layers. As for the deployment of SVOs, the users can choose among a list of applications in the relevant repository, and the APP deployer provides for putting the source code in the user cloud space. As discussed for the SVOs, the user is in charge of the running and storage costs.

4 Lysis Prototype Development

To implement Lysis, we chose the Google App Engine (GAE) PaaS as container at the different architectural levels. The choice was guided by the fact that any user is provided with an user-friendly environment where to instantiate 25 free web services. This fact is very important to get an initial population of SVOs allowing people to try this new IoT environment. Furthermore, GAE comes with key useful APIs [25]: Search API and Maps API. The former allowed us to implement on each SVO a template repository of friends by means of document representation enabling full-text search through the social graph; the latter allowed us to use an uniform repository of locations which are needed for the social relations CWOR and CLOR, which rely on information about objects positions. Specifically, the Search API provides a model for indexing documents that contain structured data and supports text search on string fields. The documents and indexes are stored in separate datastores optimized for search operations. It does not fit applications with large result sets; however, it is used in our social environment, where there is a separate database instance for each SVO, with a limited size given by the number of friends. The Search API are principally used during SVO search, according to which an SVO looks for friends that may provide a target service in its local database. The service is available for tests here[2]. It is still under development, but it is already capable to create relationships among registered devices, to manage device resources and to carry out a full text SVO resource search.

[2] http://www.lysis-iot.com.

5 Conclusions

In this paper we have presented the IoT platform called Lysis, which presents four major features: it has been designed to exploit the PaaS service model; the Social Virtual Object is a key element; user data and applications are stored and executed in the user cloud space; re-usability of templates and applications is put forward. The implementation of the platform on the Google App Engine PaaS showed that this solution was greatly facilitated by the available API for semantic search and localization. Other important aspects remain to be explored: the issue of task allocation among the real objects and the virtual counterparts through runtime code injection into the real devices; deployment of the SVO in distributed cloud (edge/fog clouds) to follow the physical device to reduce latency; large use-cases deployments.

Acknowledgment. This work has been partially supported by the project "Platform for the deployment of distributed applications in Vanets and WSNs" funded by Telit Communications SpA and by the project SocialMobility, "P.O.R. FESR 2007-2013 Regione Sardegna - Asse VI Competitività 6.2.1 a", CUP F25C10001420008

References

1. Da Xu, L., He, W., Li, S.: Internet of Things in industries: a survey. IEEE Trans. Ind. Inform. **10**(4), 2233–2243 (2014)
2. Foteinos, V., Kelaidonis, D., Poulios, G., Vlacheas, P., Stavroulaki, V., Demestichas, P.: Cognitive management for the Internet of things: a framework for enabling autonomous applications. IEEE Veh. Technol. Mag. **8**(4), 90–99 (2013)
3. ICORE Project: Deliverable 2.1 (2012). http://www.iot-icore.eu
4. COMPOSE Project: Collaborative open market to place objects at your service (2012). http://www.compose-project.eu
5. IoT-A Project. Deliverable 1.4 (2012). http://www.iot-a.eu
6. FIWARE Project (2011). http://www.fiware.org/
7. Glikson, A.: FI-WARE: core platform for future internet applications. In: Proceedings of the 4th Annual International Conference on Systems and Storage (2011)
8. Soliman, M., Abiodun, T., Hamouda, T., Zhou, J., Lung, C.H.: Smart home: integrating internet of things with web services and cloud computing. In: IEEE 5th International Conference on Cloud Computing Technology and Science (CloudCom), vol. 2, pp. 317–320. IEEE (2013)
9. Perera, C., Zaslavsky, A., Christen, P., Georgakopoulos, D.: Sensing as a service model for smart cities supported by internet of things. Trans. Emerg. Telecommun. Technol. **25**(1), 81–93 (2014)
10. Amaral, L.A., Hessel, F.P., Bezerra, E.A., Corra, J.C., Longhi, O.B., Dias, T.F.: eCloudRFIDA mobile software framework architecture for pervasive RFID-based applications. J. Netw. Comput. Appl. **34**(3), 972–979 (2011)
11. Tao, F., Cheng, Y., Da Xu, L., Zhang, L., Li, B.H.: CCIoT-CMfg: cloud computing and Internet of things-based cloud manufacturing service system. IEEE Trans. Ind. Inform. **10**(2), 1435–1442 (2014)

12. Yang, G., Xie, L., Mantysalo, M., Zhou, X., Pang, Z., Da Xu, L., Zheng, L.R.: A health-IoT platform based on the integration of intelligent packaging, unobtrusive bio-sensor, and intelligent medicine box. IEEE Trans. Ind. Inform. **10**(4), 2180–2191 (2014)
13. Fang, S., Da Xu, L., Zhu, Y., Ahati, J., Pei, H., Yan, J., Liu, Z.: An integrated system for regional environmental monitoring and management based on internet of things. IEEE Trans. Ind. Inform. **10**(2), 1596–1605 (2014)
14. Miorandi, D., Sicari, S., De Pellegrini, F., Chlamtac, I.: Internet of things: vision, applications and research challenges. Ad Hoc Netw. **10**(7), 1497–1516 (2012)
15. Sun, E., Zhang, X., Li, Z.: The internet of things (IOT) and cloud computing (CC) based tailings dam monitoring and pre-alarm system in mines. Saf. Sci. **50**(4), 811–815 (2012)
16. He, W., Yan, G., Da Xu, L.: Developing vehicular data cloud services in the IoT environment. IEEE Trans. Ind. Inform. **10**(2), 1587–1595 (2014)
17. Li, F., Vgler, M., Claeens, M., Dustdar, S.: Efficient and scalable IoT service delivery on Cloud. In: IEEE Sixth International Conference on Cloud Computing (CLOUD), pp. 740–747. IEEE (2013)
18. Gubbi, J., Buyya, R., Marusic, S., Palaniswami, M.: Internet of Things (IoT): a vision, architectural elements, and future directions. Future Gener. Comput. Syst. **29**(7), 1645–1660 (2013)
19. Mendes, P.: Social-driven internet of connected objects. In: Proceedings of the Interconn, Smart Objects with the Internet Workshop (2011)
20. Ding, L., Shi, P., Liu, B.: The clustering of internet, internet of things and social network. In: 3rd International Symposium on Knowledge Acquisition and Modeling (KAM), pp. 417–420. IEEE (2010)
21. Atzori, L., Iera, A., Morabito, G.: Siot: Giving a social structure to the internet of things. IEEE Commun. Lett. **15**(11), 1193–1195 (2011)
22. Atzori, L., Iera, A., Morabito, G., Nitti, M.: The social internet of things (siot)when social networks meet the internet of things: concept, architecture and network characterization. Comput. Netw. **56**(16), 3594–3608 (2012)
23. Girau, R., Nitti, M., Atzori, L.: Implementation of an experimental platform for the social internet of things. In: 2013 Seventh International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), pp. 500–505. IEEE (2013)
24. Nitti, M., Girau, R., Atzori, L.: Trustworthiness management in the social internet of things. IEEE Trans. Knowl. Data Eng. **26**(5), 1253–1266 (2014)
25. Google: Google search api. https://cloud.google.com/appengine/docs/java/search/