Enabling actor model for crowd sensing and IoT

Conference or Workshop Item

Accepted Version

Sánchez, D. D., Sherratt, S., Arias, P., Almenarez, F. and Marín, A. (2015) Enabling actor model for crowd sensing and IoT. In: 2015 IEEE International Symposium on Consumer Electronics (ISCE), 24-26 June 2015, Madrid, Spain. Available at http://centaur.reading.ac.uk/43187/

Full-text of article

Title:	Enabling Actor Model for Crowd Sensing and IoT

Publication:	2015 IEEE International Symposium on Consumer Electronics (ISCE15)
Article URL:	http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=7177779
Article DOI:	10.1109/ISCE.2015.7177779
Conference Location:	Madrid, Spain
Conference Date:	24-26 June 2015

Authors:

Daniel Diaz Sánchez,
School of Systems Engineering, The University of Reading, UK, and Telematic Engineering Department Universidad Carlos III de Madrid, Leganés, Spain

R. Simon Sherratt
School of Systems Engineering, The University of Reading, UK

Patricia Arias,
Telematic Engineering Department, Universidad Carlos III de Madrid, Leganés, Spain

Florina Almenarez
Telematic Engineering Department, Universidad Carlos III de Madrid, Leganés, Spain

Andrés Marín
Telematic Engineering Department, Universidad Carlos III de Madrid, Leganés, Spain

This work was partially funded by the Ministerio de Educacion, Cultura y Deporte through the Jose Castillejo grant (CAS14-00364) and the project EMRISCO (TEC2013-47665-C4-4-R)

Abstract

The cloud is playing a very important role in wireless sensor network, crowd sensing and IoT data collection and processing. However, current cloud solutions lack of some features that hamper the innovation a number of other new services. We propose a cloud solution that provides these missing features as multi-cloud and device multi-tenancy relying in a whole different fully distributed paradigm, the actor model.

Keywords

crowd sensing; Internet of Things; actor model; cloud computing; big data

I. Introduction

Wireless Sensor Networks (WSN) have been the forefather of the technology around crowd sensing [1] and Internet of Things (IoT) paradigms, serving a huge range of applications in commercial, industrial, environmental and eHealth fields. Initially, WSN research concentrated in energy saving and harvesting, routing and transmission favoring sensor self-regulation and enlarging their life-time. As long as the field developed, the increasing amount of data collected and process by such systems became a real problem so adequate backend server technology was developed to cope with that, bringing sensor grid concept. Recently, sensor-cloud [2] has been adopted as a more flexible alternative. As opposed to grid, that requires tailored developments, the cloud has a general purpose, is widely available, easy to use and cheap.

Applications in IoT and crowd sensing scenarios generate also a huge amount of information to be processed. Nowadays, manufactures are developing small general purpose boards with integrated network interfaces, processing power and storage where to attach sensors and actuators. These boards are typically shipped with a built-in operating system, libraries and the software development kit for connecting things to their cloud services, providing so a customizable solution for IoT or sensing in a single bundle. However, current solutions are tightly coupled to proprietary cloud software (centralized), and users are captive of a given provider avoiding multi-cloud. We introduce our proposal for a better sensing and IoT, not proposing yet-another-framework but relying in a different paradigm compatible with current cloud: the actor model.

II. Sensor Cloud and IoT

Sensors and things, are devices distributed in a given area for the purpose of collecting data about their surroundings and actuating accordingly. Processing the data collected and providing an outcome requires the use of grid and cloud. Despite both grid and cloud concepts can be sometimes confused, there are several differences. Grid is a decentralized model in which the computation could occur in different administrative domains whereas cloud use to be a centralized model (single provider). In both cases, several computing resources are joined together for accomplish a computation problem but in cloud computing, resources are no limited to computing and storage and provides virtualization.

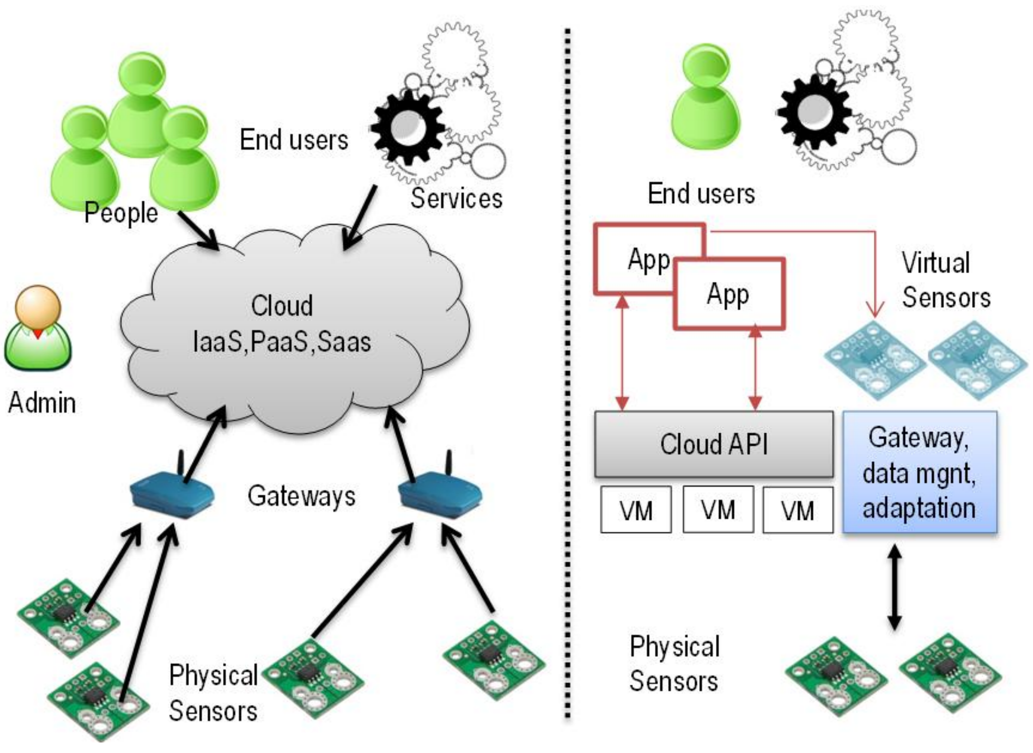

Fig. 1. Sensor cloud structure

On top of virtualization (IaaS), cloud providers have developed APIs to build scalable complex application disguising the underlying complexity (PaaS or SaaSI). Beyond collecting and processing data, the cloud should enable information sharing on big scale with the applications running on that cloud and among users. Fig.1 shows the common architecture of a sensor-cloud system.

Physical devices, by means of their build in software, send data to the cloud through gateways, which can be also used for adapting device data to the cloud. The data is stored and applications running on the cloud can process the data and present the results to end users or feed other application with the outcome. Cloud computing resources are shared in a multitenant environment, however, physical devices, cannot be easily shared in the same way. Due to that, devices can be virtualized so they can be created, modified and destroyed on demand [3]. Despite this kind of device virtualization provides some flexibility, the functionality of the virtual sensor is limited, as well as the cloud application, by the specific API of the cloud provider. This constitutes one of the major problems in cloud computing known as data lock-in. It also prevents the data to be used in other clouds (multi-cloud). These limitations in API and virtualization affect the data processing since it is completely performed in the cloud. Devices encapsulate data into messages and sent them to the cloud. Then, data is stored, filtered (curation) and finally processed. Despite virtual sensors can be used to perform a pre-processing, that is entirely handled by the cloud, being interesting in many cases to do it on the device. For instance, triggering alarms immediately upon detection of events, or filtering unnecessary data samples.

III. Actor model for Cloud Sensor and IoT

A. Requirements

Our system requires using device processing power to execute small applications called actors. The very best improvement of cloud computing over other distributed computing systems is the easiness of development and deployment. Applications running on our system should be easy to develop and deploy; in fact, we pursue to keep exactly the same model that would be used in a data center despite it will be executed elsewhere. Moreover, we are conscious of the limitations of the device hardware, so actors should consume very few resources. Among the most interesting requirements are multi-cloud and device multi-tenancy. Multi-cloud allows several cloud providers to communicate with the actor system running on devices, so several user applications running on different cloud providers can share the same data, preventing the data lock-in problem. When it comes to device multi-tenancy, shared devices may execute actors from different users. This enables on-device virtualization in addition to the device virtualization in the cloud. Unlike existing cloud systems, our on-device actor system allows to manipulate data at the device and in the cloud as well, in a per application or user basis. Other new feature worth to mention is the support of device disconnection and itinerancy, and asynchronous communication. Current solutions require devices to be connected to the cloud to perform every single operation. For instance, SDKs for indoor positioning beacon systems that require connection to the cloud to resolve the identifiers of the beacons (ensuring their hardware cannot be used with other providers) whereas it could be more efficient and robust in some cases to fetch the geo-localization information once for a building and work offline after that. The asynchronous communication enables the use of carry and forward protocols, simplifies the execution of concurrent operations and allows optimizing the network resources by grouping messages, filtering or scheduling deliveries.

B. The actor model

The actor model is a well-known mathematical model of concurrent large scale computation. Actors are the universal primitives of computation. Applications based on actor model begin with an actor who can create others. An actor is very simple; it has a thread and a mailbox where it can receive messages and can keep some state. The actor behavior is controlled by the internal state and the messages received. Actors run isolated helping to keep data private in a multitenant environment. The Fig. 2 shows how an application can be partially instantiated in different providers and communicates with its corresponding actors running on devices. Thus, this makes possible multi-cloud and device multi-tenancy. Actors are very light weight single-threaded applications but in a multitenant environment they may bring down the device or drain the battery if it not controlled. Our system provides a token bucket system to control the usage actors make of the available device resources.

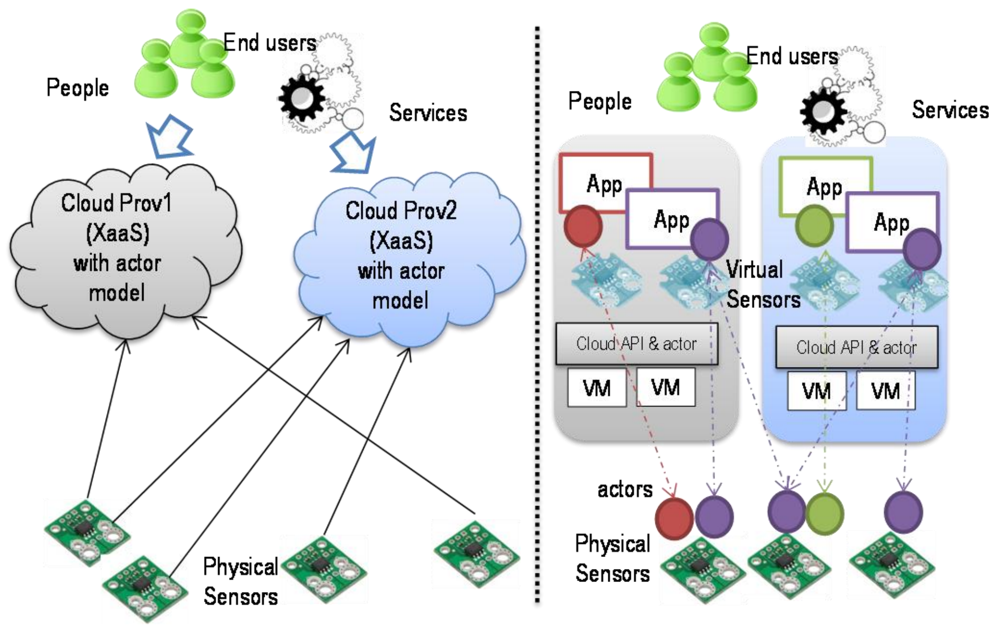

Fig. 2. Actor model integration for multi-cloud and device multitenancy

The communication is handled via message passing over virtually any protocol. We rely on class serialization being possible to implement any serialization mechanism. The system guarantees a single incoming message is processed at a given time, requiring no further synchronization mechanisms. The system manages device mobility by implementing a clearinghouse or repository. Every living actor system running on a device registers or updates their addressing information and resource availability when they come to life or change to other networks. The clearinghouse is an actor that can be instantiated in several clouds providing redundancy.

IV. Implementation and Conclusions

There are several actor model implementations, from Erlang to the AKKA framework based on Scala and Java. We are constantly improving a mockup using the open source AKKA framework so actors can be programmed in Java or Scala and using Raspberry Pis with a bunch of sensors and actuators. Every board runs an AKKA framework that can load new actors on demand when instructed by the clearinghouse actor. We are collecting data using directly AKKA and also using public clouds. We are currently developing device virtualization on top of akka using OpenvSwitch. The cloud has a very important role in the upcoming IoT and crowd sensing scenarios, but current solutions limit the way data is managed and will hamper the development of new cross-domain distributed services in scenarios in which cooperation is critical. We believe actor model is a perfect complement to enable new features in these environments due to their low consumption, easiness of use and robustness.

References

[1] Lane, N.D.; Miluzzo, E.; Hong Lu; Peebles, D.; Choudhury, T.; Campbell, A.T., "A survey of mobile phone sensing," Communications Magazine, IEEE , vol.48, no.9, pp.140,150, Sept. 2010

[2] Atif Alamri, et. Al., “A Survey on Sensor-Cloud: Architecture, Applications, and Approaches,” International Journal of Distributed Sensor Networks, vol. 2013,

[3] Yuriyama, M.; Kushida, T., "Sensor-Cloud Infrastructure - Physical Sensor Management with Virtualized Sensors on Cloud Computing," Network-Based Information Systems (NBiS), Sept. 2010