2015 16th IEEE International Conference on Mobile Data Management

iCOMOT - A Toolset for Managing IoT Cloud Systems

Hong-Linh Truong, Georgiana Copil, Schahram Dustdar, Duc-Hung Le, Daniel Moldovan, Stefan Nastic
Distributed Systems Group, Vienna University of Technology
E-mail: {truong, e.copil, dustdar, d.le, d.moldovan, s.nastic}@dsg.tuwien.ac.at

Abstract

Developing and operating IoT cloud systems require novel features for deploying, controlling, monitoring and testing both IoT units and cloud services in an integrated environment spanning different infrastructures. In this paper, we demonstrate $iCOMOT$ – a novel toolset offering these features. Using $iCOMOT$ we can perform various activities, such as dynamically reconfiguration of sensors, communication protocols, and cloud services in an elastic manner, suitable for testing and assuring quality of IoT cloud systems configurations. We will demonstrate our $iCOMOT$ with a real-world predictive maintenance case study.

1. Introduction

Advances in Internet of Things (IoT) and Cloud computing have fostered the development of several frameworks for the so-called IoT cloud or cloud-based M2M platforms [1]. In such platforms, sensors gather and send data to cloud services to provide large-scale, near-real time data for several different application domains, such as smart cities, building management, and logistics. In developing and operating IoT cloud systems/applications atop such platforms, one of the great challenges is how to manage sensors, gateways, communications, and services and coordinate their activities to support the provisioning of suitable, reliable data and services for further data analytics. To answer this challenge, we need to carry out several tasks which require novel toolsets as these tasks are complex, error-prone and time-consuming.

In this work, we will demonstrate $iCOMOT$ – a set of tools and services that simplify the management of such sensors, gateways and services. At the edge of the IoT cloud system, $iCOMOT$ employs the concept of software-defined IoT units [2] to abstract and implement sensors and software components atop gateways for data, control and connectivity functions. At the cloud data center, $iCOMOT$ considers all software components and resources as elastic service units [3]. Based on that, $iCOMOT$ can support the IoT cloud developer and provider to carry out the following activities for their IoT cloud systems:

- Deployment and configuration of software-defined IoT units at the edges and cloud services at data centers of IoT cloud platforms
- Governance and reconfiguration of software-defined IoT units capabilities at the edges
- Elasticity analytics and control of gateways and cloud services based on changes in software-defined IoT units in a coordinated fashion

In this work, we will illustrate these features with our realistic scenarios for smart city management in the context of predictive maintenance problems. The rest of this paper is structured as follows: Section 2 discusses our motivating scenarios and required features. Section 3 explains features of $iCOMOT$. We conclude this work in Section 4.

2. Motivating Scenario

Our motivating scenario is in the domain of facility management in the cloud. In our scenario, a predictive maintenance company buys several cloud services in data centers for handling and managing near-real time sensoring data. Such sensoring data will be provided for other analytics, such as predicting possible maintenance problems of equipment (e.g., chillers) being sensed. The sensoring data are provided by various "Things" interfacing to different equipment and facility systems (such as chillers and electricity systems). Sensoring data are sent to the cloud services via several gateways, which are composed of lightweight hardware and software components deployed in the edge, acting as intermediate nodes between the cloud and the Things. Things connect to gateways via different protocols (such as CAN-Bus[1] and LonWorks[2]) and gateways connect to the cloud via different protocols (such as CoAP[3], MQTT[4] and specific REST/HTTP-based protocols).

In this work, we are interested in the scenario where the predictive maintenance company needs to manage and control several sensors and services in order to provide suitable data for data analytics of chillers in a city. To this end, the company needs several features to test and manage its end-to-end IoT cloud system consisting of sensors, gateways and cloud services and being deployed across different IoT and cloud infrastructures. For example, several sensors can be deployed but not all of them would be activated at the beginning as well as new sensors can be added into the existing IoT cloud system. Several sensors may be configured with predefined frequencies for reading data from equipment as well as gateways can be configured with particular communication protocols. However, at runtime, due to predictive maintenance analytics, sensors can be reconfigured with different reading frequencies, leading to different amount of data generated.

1. http://en.wikipedia.org/wiki/CAN_bus
2. http://en.wikipedia.org/wiki/LonWorks
3. http://coap.technology/
4. http://mqtt.org/

978-1-4799-9972-9/15 $31.00 © 2015 IEEE
DOI 10.1109/MDM.2015.65

Thus, as a consequence, we need to reconfigure gateways and their cloud connectivity by setting suitable communication protocols. Furthermore, cloud services at the data centers need to be controlled to deal with changing incoming sensor data. As all of these activities are carried on demand and elastic, the predictive maintenance company needs suitable tool features for automatic and manual deployment, control and governance of the above-mentioned IoT cloud system.

3. iCOMOT – IoT Cloud Control, Monitoring and Testing

3.1. iCOMOT Overview

Figure 1 describes the above-mentioned IoT cloud system and our $iCOMOT$ main tools and services. $iCOMOT$ is built atop our services in COMOT [3] – designed for cloud services – and – IoT units and GovOps [2], [4] – designed for IoT. Services in COMOT have been extended to support also features required for testing and managing IoT cloud systems configurations, such as coordinated deployment, analytics and control across platforms.

Overall, the *Deployment and Configuration* will perform cloud service deployments as well as IoT units deployment in gateways. For testing, *Deployment and Configuration* will deploy emulated gateways and sensors with realistic sample datasets. *IoT Governance* is designed for supporting governing the operation of IoT service units [2], [4], whose capabilities can be changed via software-defined APIs. The *Monitoring and Analytics* is used to provide high-level elasticity metrics and dependencies (such as, for performance and cost) [5]. *Elasticity Control* performs elasticity strategies for cloud services and IoT units in gateways to assure that the service will operate properly under elastic workload. Therefore, it also utilizes information from *Monitoring and Analytics* and invokes *Deployment and Configuration* and *IoT Governance*

In this work, we will show $iCOMOT$ features supporting the challenges outlined in Section 2 by simplifying the tasks of the developer and operator. For experiments, we will emulate gateways and sensors based on our industrial settings, while cloud services will be based on common services available in the cloud.

3.2. Deployment and Configuration

While deployment of cloud services has been intensively researched, deployment of elastic IoT units has just been investigated recently. $iCOMOT$ supports both types of deployment in an integrated manner and provides a mechanism to configure different deployments of different parts of the IoT cloud system to work together. For example, one can deploy IoT units as sensors separated from the deployment of cloud services and emulated gateways. In $iCOMOT$, we use TOSCA [6] as a means to describe topologies of IoT units or cloud services to be deployed and we support different underlying low-level deployment technologies in order to deploy various types of software, such as virtual machines and cloud services in the data centers and sensors, communication middleware and OS containers in the gateways. Both programmable APIs and GUI are provided for large-scale deployments. For example, Figure 2 shows an example of different views on sensor topology and its deployment status.

Another feature is that to allow the activation and deactivation of sensors for specific data analytics. Therefore, our deployment and configuration allows us to manage states of sensors and query information about sensors (such as the location and types of data a sensor provides) so that we can activate and deactivate sensor instances, depending on analytics requirements from the cloud. Furthermore, such a detailed information can also be useful for governance processes that we will discuss in the next section.

3.3. Governing IoT Units

At runtime, several reasons require us to govern the operation of IoT units, such as sensors, gateways, and their communications. For example, we might need to change the data reading sampling rates as well as communication protocols when we need more fine-grained data in order to support a reliable analytics. Such activities are performed through the concept of GovOps. The core feature is that every IoT unit has capabilities that can be changed at runtime via software-defined APIs. Thus, when we need to reconfigure IoT units, we can invoke capabilities with suitable parameters.

In $iCOMOT$ this is achieved through the implementation of governance processes, core services to execute capabilities of IoT units, and the implementation of software-defined capabilities of IoT units. Listing 1 shows an example of a governance process for reconfiguring the communication protocol from CoAP to MQTT for a unit deployed within a gateway. In this process, we call different actions, each invokes a capability of the unit. When an action is executed, our runtime services will change the unit based on different mechanisms. For example, when the unit implements its software-defined capabilities via listening runtime variables, the runtime services could change capabilities by stopping the unit and changing variables and restarting the unit. An important feature is that these governance capabilities are exposed for elasticity actions that we will illustrate in the next section.

Listing 1. Example of governance process

```
invokeCapability(){
...
}

#Start GovOps process with the normal
    operation
invokeCapability family"familycChangeProto
    family/familychangefamily/familyaruments
    family?familyargfamily=familycoapfamily" 1

#Run coapClient for a few minutes
```

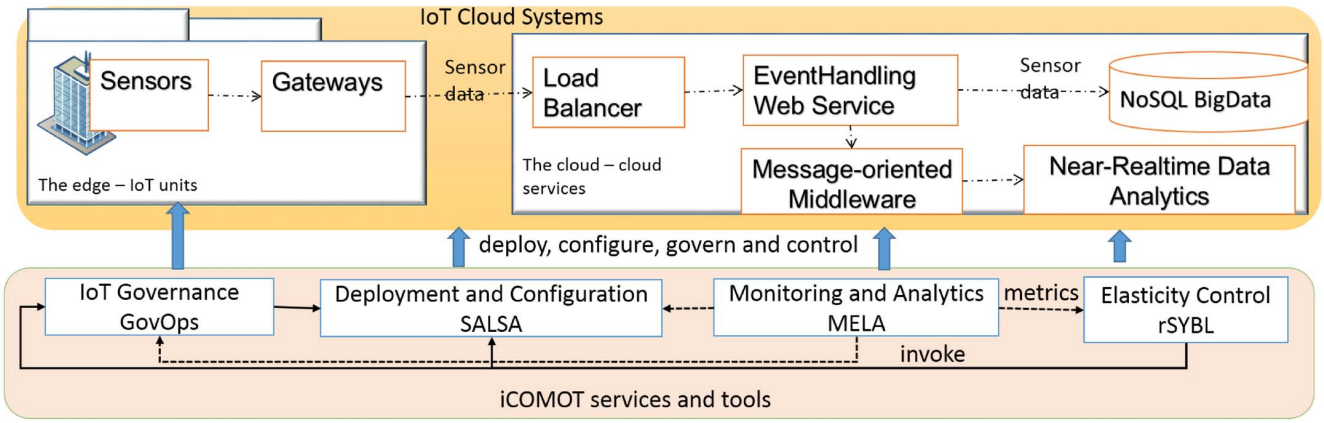
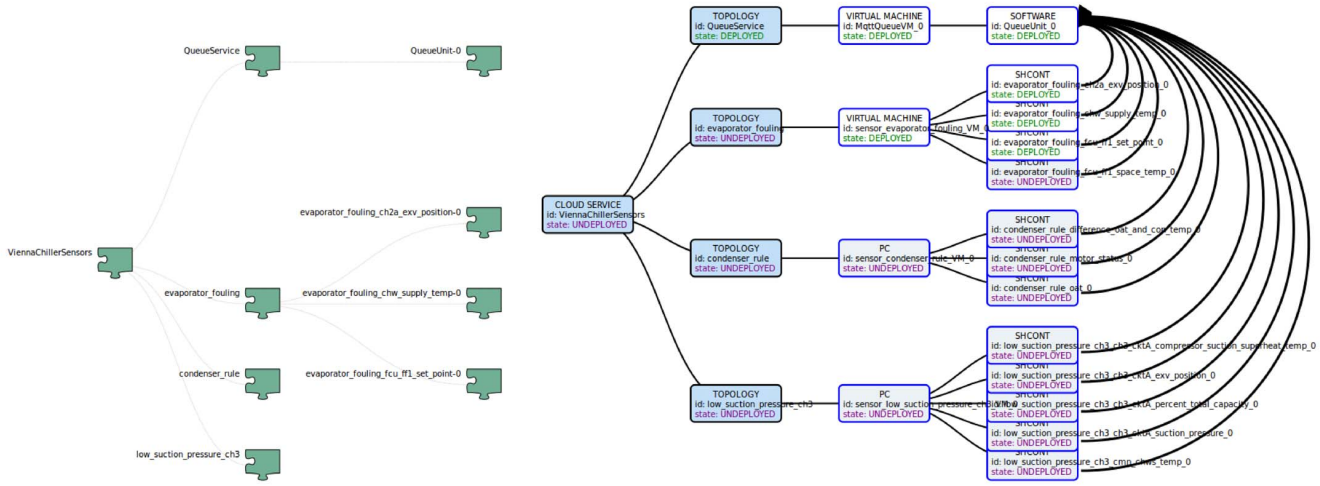


Fig. 1. Overview of iCOMOT and its services and tools



(a)Example of a topology of sensors (b)Example of deployment status of the topology of sensors in (a)

Fig. 2. Examples for a topology of sensors

```
#First just run coap client normally
/home/ubuntu/coapClient.sh 1 6

#Second start the mqtt
invokeCapability family"familycChangeProto
    family/familychangefamily/familyaruments
    family?familyargfamily=familymqttfamily" 2

#Third invoke GovOps process
#and start coapClient
invokeCapability cManager/list
invokeCapability cManager/list
invokeCapabilityMixed family"
    familycChangeProtofamily/familychange
    family/familyargumentsfamily?familyarg
    family=familycoapfamily"

#Fourth just run coap client
/home/ubuntu/coapClient.sh 1 6
```

3.4. Elasticity of Data Cloud Services

When IoT units are deployed and running, several cloud services will receive data and handle them accordingly. Such cloud services have pre-defined configurations (such as, only a few data nodes are used for storing data in Cassandra) and elasticity strategies are specified to deal with different situations (e.g., increase the buffer size of the communication protocol when there is an increasing amount of data). In *iCOMOT* this is achieved by using SYBL and its runtime system [7]. Depending on the monitoring information, corresponding elasticity strategies, which can be specified at a very high-level, will be invoked when the workload of cloud services change (e.g., as a consequence of changing data reading frequencies and activating sensors). To support elasticity control, detailed analytics must be provided in an end-to-end manner. Another feature is that elasticity strategies can be also applied to IoT units. In this case, the elasticity

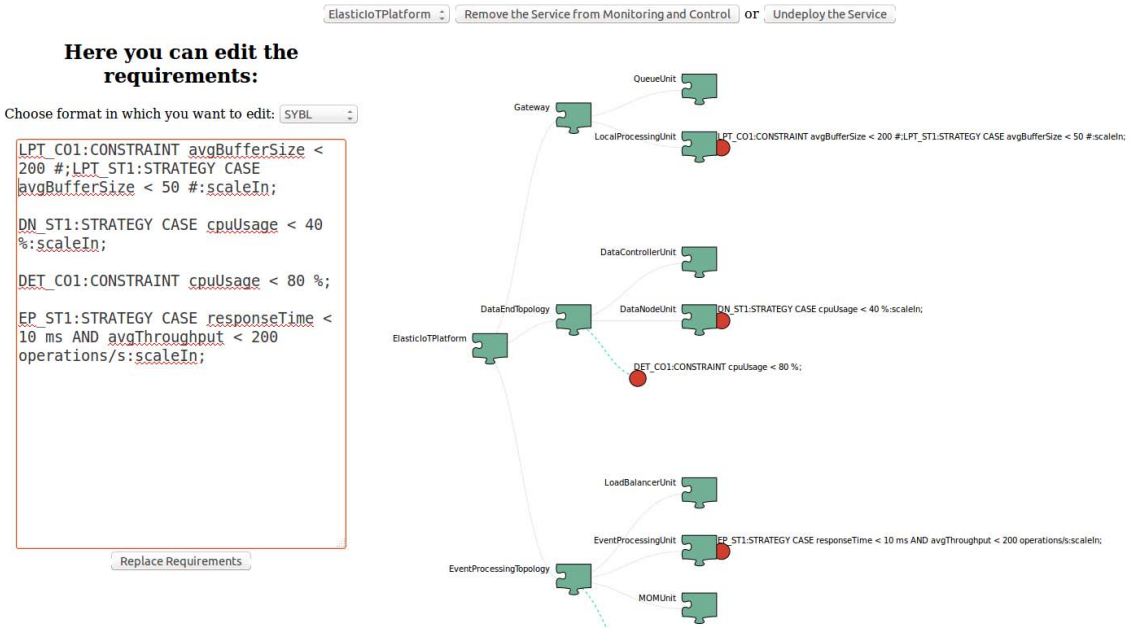


Fig. 3. Example of using UI to manage and change elasticity constraints and strategies.

control will invoke IoT governance processes when executing elasticity strategies for IoT units. This two-ways of interaction enables us to carry out the elasticity control in a coordinated manner across the whole IoT system. The control of elasticity will be done automatic but in many cases, the developer and the provider want to change, they can use an interactive mode (see Figure 3).

4. Conclusions and Future Work

In this work, we outlined features of $iCOMOT$ for controlling, monitoring and testing IoT cloud systems consisting both IoT units and cloud services. A set of features are provided from different services can be used to simplifying support activities of developers and providers in deploying and reconfiguring their systems to enable data provisioning for reliable data analytics. Currently, we focus on further integration and testing of $iCOMOT$, of which tools and services are published under open source and available freely.

Acknowledgment

This paper is partially supported by the European Commission in terms of the CELAR FP7 project (FP7-ICT-2011-8 #317790) and the H2020 U-Test project.

References

[1] A. Botta, W. de Donato, V. Persico, and A. Pescape, "On the integration of cloud computing and internet of things," in *Future Internet of Things and Cloud (FiCloud), 2014 International Conference on*, Aug 2014, pp. 23–30.

[2] S. Nastic, S. Sehic, D. Le, H. L. Truong, and S. Dustdar, "Provisioning software-defined iot cloud systems," in *2014 International Conference on Future Internet of Things and Cloud, FiCloud 2014, Barcelona, Spain, August 27-29, 2014*, 2014, pp. 288–295. [Online]. Available: http://dx.doi.org/10.1109/FiCloud.2014.52

[3] H. L. Truong, S. Dustdar, G. Copil, A. Gambi, W. Hummer, D. Le, and D. Moldovan, "Comot - A platform-as-a-service for elasticity in the cloud," in *2014 IEEE International Conference on Cloud Engineering, Boston, MA, USA, March 11-14, 2014*. IEEE, 2014, pp. 619–622. [Online]. Available: http://dx.doi.org/10.1109/IC2E.2014.44

[4] S. Nastic, M. Voegler, C. Inziger, H.-L. Truong, and S. Dustdar, "rtgovops: A runtime framework for governance in large-scale software-defined iot cloud systems," in *The 3rd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering*.

[5] D. Moldovan, G. Copil, H. L. Truong, and S. Dustdar, "MELA: monitoring and analyzing elasticity of cloud services," in *IEEE 5th International Conference on Cloud Computing Technology and Science, CloudCom 2013, Bristol, United Kingdom, December 2-5, 2013, Volume 1*. IEEE, 2013, pp. 80–87. [Online]. Available: http://dx.doi.org/10.1109/CloudCom.2013.18

[6] T. Binz, G. Breiter, F. Leymann, and T. Spatzier, "Portable cloud services using TOSCA," *IEEE Internet Computing*, vol. 16, no. 3, pp. 80–85, 2012. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/MIC.2012.43

[7] G. Copil, D. Moldovan, H. L. Truong, and S. Dustdar, "SYBL: an extensible language for controlling elasticity in cloud applications," in *13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing, CCGrid 2013, Delft, Netherlands, May 13-16, 2013*. IEEE Computer Society, 2013, pp. 112–119. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/CCGrid.2013.42