International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 06 Issue: 04 | Apr 2019 www.irjet.net p-ISSN: 2395-0072

# Testing Uncertainty of Cyber-Physical Systems in IoT Cloud Infrastructures: Combining Model-Driven Engineering and Elastic Execution

**Ms. A. Sivasankari., Mrs. N. Sindhuja., Mr .D.B. Shanmugam**

*[1]Head of the department, Department of Computer science, DKM College for Women (Autonomous), Vellore.*
*[2]Research scholar,Department of Computer Science, DKM College for Women (Autonomous), Vellore, TamilNadu.*
*[3]Headof the Department(MCA), Sri Balaji Chockalingam Engineering College, Arani, TamilNadu.*

---

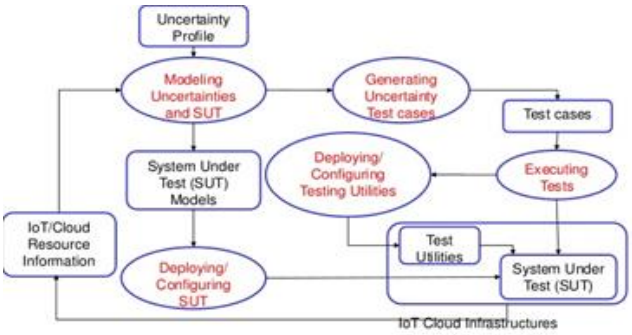
**Abstract - -**Today's cyber-physical systems (CPS) span IoT and cloud-based datacenter infrastructures, which are highly heterogeneous with various types of uncertainty. Thus, testing uncertainties in these CPS is a challenging and multidisciplinary activity.We need several tools for modelling, deployment, control, and analytics to test and evaluate uncertainties for different configurations of the same CPS.In this paper, we explain why using state-of-the art model-driven engineering (MDE) and model-based testing (MBT) tools is not adequate for testing uncertainties of CPS in IoT Cloud infrastructures. We discuss how to combine them with techniques for elastic execution to dynamically provision both CPS under test and testing utilities to perform tests in various IoT Cloud infrastructures. Ccs Concepts(1) Computing methodologies→Model development and analysis;(2) Computer systems organization.

**KEYWORDS:** Testing, Elasticity, Uncertainty, IoT, Cloud, MDE, MBT

## 1. INTRODUCTION:

Cloud computing is adopting significance in the current technology. Due to the advent in the communication technology, any objects can get connected with any other objects. This is the era of where the things speak instead of livings humans. This technology has been termed as Internet of Things (IoT). For IoT to speak, there is a need of powerful essential network infrastructure that is Internet. This is in turn prone to physical threats in the cloud network. IoT is an integrated part of future Internet and can be defined as a dynamic global network infrastructure with self-configuring capabilities based on standard and interoperable communication protocols where physical and virtual "things" have identities, physical attributes, and virtual personalities and use intelligent interfaces, and are seamlessly integrated into the information network .

The continuous growth of the urban population has generated a tremendous expansion of our cities.Nowadays, indeed, more than 50 % of the world's population is living urban, and they estimate that it will reach 70% by the year 2050. Therefore, cities need to be ready to accommodate this huge amount of citizens and to face new challenges like traffic congestion, air pollution, waste management and so on. Majority of the IoT companies included are Amazon, CISCO, MS Azure, Salesforce, Oracle and much more. IoT is the integration of multiple heterogeneous networks. It is difficult to establish the junction of relationship as the relationship of trust between nodes that are constantly changing. Security has become a serious aspect of IoT because of the inclusion of the latest technologies and various applications to enormous users.




**1. IOT Cloud model**

## 2. IOT AND CLOUD INTEGRATION MODELS:

### 2.1 Developers Of Iot Cloud Platforms Can Have Varying Goals:

Goal 1: developers might need only to develop IoT elements (e.g., sensors, actuators and gateways) for a customer and to connect these IoT elements to existing cloud services. In this case, they might just want to develop and test a set of sensors that can be deployed into certain gateways sending the data to public cloud services.

Goal 2: developers focus on only cloud services at data centers that serve IoT elements. Typically they focus on a set of complex cloud services, e.g., for data storage, complex event processing, and data analytics.

Goal 3: developers want to design and test a complete IoT cloud platform for a customer. They, therefore, focus on both IoT elements and cloud services and on how to coordinate them in a united view for the customer. Numerous works support the development of either the IoT part or the cloud

services for IoT . However, there is a lack of tools and discussions for the development and operations of the last goal.

Goal 4. We focus on supporting developers to concentrate on Goal 3, which is complex but of paramount importance for several customers, e.g., predictive maintenance of equipment, on-demand crowd sensing for safe cities, and sports events. For such platforms, we must deal with deferent engineering principles outlined.

### 2.2 Development In Distributed Iot And Cloud Systems

As the IoT cloud platform is complex, it is expected that during the development, components of the platform must be able to deploy in multiple IoT and cloud systems.

For this, we must have a set of connectors allowing the developer access to clouds and IoT systems so that the developer can deploy testing infrastructures and run tests across clouds and IoT septic systems. We also need to deal with different mechanisms of controlling virtual resources and different performance settings (e.g., expected time for allocating a resource, ex-pected performance for each resource).

As discussed in the related work most tools either enable IoT deployment or cloud deployment; even many industrial cloud systems support IoT and cloud and enable their integration, these systems support Goal 1 and Goal 2.Moreover, at runtime, both IoT elements and cloud services need to be controlled, monitored, and analyzed in a coordinated manner.

While throughout the development lifecycle the developer would need mechanisms to emulate sensors and gateways in the heterogeneity of different clouds, in a production environment we have to do an end-to-end control of cloud services and gateways deployed across IoT networks and clouds.

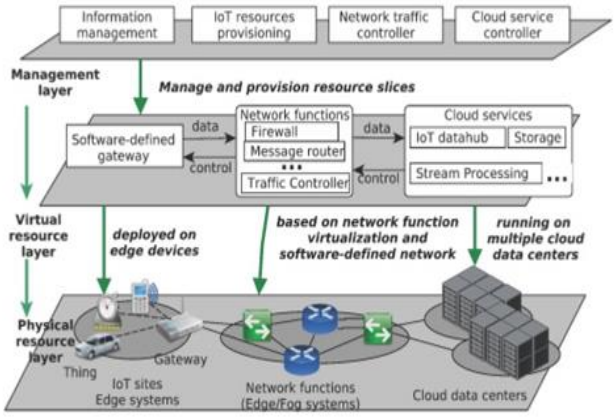



**Fig 2:IOT AND CLOUD INTEGRATION MODELS**

This has to be done in a uniform manner, and most of the times control on one end of the IoT cloud platform would the control on the other end (e.g., deploying new data processing services on the gateway would change the characteristics of the load on cloud software services).As discussed in the related work, this feature is missing in most toolsets.

## 3 .OVERVIEW OF ICOMOT

To support Goal 3 in multi-IoT and -cloud environments, we design, develop and experience different tools and engineering actions to address two different main issues: (I) to provide main software components which are software-defined services, deployable and to support easy ,deployment, control, and monitoring in a united manner.

**To achieve the point, we base on two main emerging research directions:**

1. software-defined IoT units: sensor and gateway components are considered as units that can be composed and controlled via software-defined APIs.

2. cloud-based elastic services: they are common cloud services by different providers. To enable the elasticity, some services will be associated with elasticity capabilities. To achieve the second point, we build a toolset to enable elasticity control developments. The describes how we leverage our existing tools and our engineering actions to develop IoT cloud platforms. Cloud providers and third-party developers can design and provide several components and services that will available through PaaS/IaaS and public repositories/marketplaces. A developer will utilize these services and components and to develop her/his own services and components.

Then can utilize various tools to support service deployment, configuration, analytics and control to test and evaluate her/his designs. To this end, we have demonstrated the iCOMOT framework to support the developer to develop and test different configurations and runtime behaviours of IoT cloud systems. Icomot (http://tuwiendsg.github.io/iCOMOT/) includes several individual research tools for configuring, deploying, monitoring and controlling IoT cloud platforms, such as SYBL, SALSA, MELA . We connect several common repositories and together with these tools to support automation of IoT cloud platform configurations deployment. Furthermore, several sensors with data sets are also provided. The following sections, we will explain main insightful engineering actions, designs and experiences.

## 4 DESIGN AND ENGINEERING ACTIONS

### 4.1 Software components for IoT cloud platforms

IoT Units and Cloud Services Repository Requirements: Existing cloud APIs already allow us to invoke suitable cloud

services at data centers and to emulate execution environments (e.g., lightweight virtual machines for emulated IoT gateways). However, they do not support the IoT side well, such as, modelling, configuring, deploying and testing different types of sensors, gateway software components, and libraries for cloud connectivity.

**Solutions:** To model and capture IoT unit capabilities and configuration with software artifact, we leverage the concept of IoT cloud units to successfully model suitable IoT resources and enable some useful behaviours, such as dynamically changing communication protocols between IoT gateways and cloud services at runtime. As IoT units can be provided by different developers and providers, we support access to different component repositories for existing IoTunits, such as gateways execution environment, virtual network routers and communication protocols for cloud connectivity.

These repositories can be leveraged by well-developed technologies, such as Docker Hub, Git-based repositories and Maven, and IoT marketplaces.

**Software Sensors Requirements:** One important type of IoT software units are sensors. For development and testing, it is crucial to have emulated sensors whose behaviour is similar to real sensors but with advanced features to allow us to easily control the sensors. An emulated sensor just takes time series datasets,e.g., obtained from industrial real systems, and simulates events by sending them to gateways. By leveraging different real datasets, we can instantiate different sensors by configuring these instances with different sample data and behaviour models. With this way, we can emulate GPS, energy consumption, temperature sensors,chiller's operational status, etc., for different types of IoT cloud
platforms.

**Solutions:** In our framework, the developer can develop her/his real sensors or emulated sensors then deposit them into the repository from which they can be deployed into IoT cloud platforms. To support rapid development of the IoT side, one important issue is to have simulated sensors as executable that can be deployed at a very large-scale in multi-cloud environments to simulate real situations, e.g., monitoring chillers in a city, in IoT cloud platforms.

We also enable the users to modify configurations while the sensors are running, for simulating different workloads or for testing their application under various out-of-the ordinary circumstances (e.g., _re at a location). The developer can also design topologies of different sensors for better management and reuse. In production scenarios of an IoT cloud platform, sensors will be physically distributed at different places, while, in simulations and tests, sensors are deployed in different VMs, OS containers, lightweight machines like Raspberry Pi, or cloud datacenters.

To enhance interoperability and reusability of possible sensor architectures, we present the topologies of sensors by well-known description languages, such as TOSCA for the deployment and control process.

Software-defined Gateways Requirements: In certain IoT cloud platforms, sensors can directly connect to cloud services. However, in our experience, very often gateways are needed as intermediate nodes for handling different types of sensoring data and connectivities. We consider and support gateways as an-other type of IoT software/hardware units. Gateways are much more complex than sensors. For example, gateways can store information and execute some lightweight components to process sensoring data in the cloudlets model.

**Solutions:** From the architecture design perspective, we develop and provision gateways functionality by using our concept of software-defined IoT units.Generally, a software-defined gateway consists of a set of dependent IoT units deployed in a virtualized environment, e.g. Centos or Docker. These IoT units are responsible for managing data streams, controls of actuators, cloud connectivity and lightweight data storage and processing.

The key point of a software defined gateway is that it enables dynamic deployment and configuration of IoT units to handle data, control and connectivity in the IoT systems. This enables the developer to implement IoT-side distributed data processing, such as pre-processing data in gateways and splitting streams, i.e., sending events to multiple cloud services. Cloud Services for IoT Requirements: At data centers, both cloud services and custom-built software components can be used for building the IoT cloud platform.Main cloud providers, such as Amazon EC2 Windows Azure.

However, it is challenging to combine and use such services in a coordinated mode with IoT elements, for example, to enable the elasticity coordination between IoT sensors and cloud services for data processing. Focusing on elastic software components, we enable developers to employ a series of o_-the-shelf software components in building their elastic platforms.

### 4.2 Deployment, Control, and Monitoring Actions

Deployment Requirements: The developer has to deploy components of IoT cloud platforms very often in order to study and test them. Generally, a deployment service will have to deal with both IoT and cloud service sides. We need the deployment of different types of services and manage from single components to the entire platform configuration at runtime. Therefore, the developer has to prepare at set of deployment artefacts in the repository including the dataset, configuration script, software artefacts. The APIs and information for accessing cloud infrastructures must also be prepared.

We witnessed that these complex tasks cannot be done by a single tool, but multiple tools and connectors to different clouds, orchestrated by a centralized service.For the application provider who just want to deploy the services or sensors, a description tool with GUI that hides the low-level information is more convenient. With an end-to-end aspect, the deployment needs to cope with different levels of deployment, including requesting cloud provider resources, configuring virtual machines, middleware and dependencies, and deploying artefacts.

**Elasticity Analytics Requirements:** For analyzing the elasticity change of the IoT cloud platform (e.g., scaling in/out cloud services and sensor instances), elasticity analytics will be deployed at different parts of the platform to provide different performance and elasticity metrics. An elastic IoT cloud platform would have elasticity requirements defined over its components, based on which intelligent controllers can analyze its behaviour and take appropriate actions. Due to the potential complexity of IoT cloud platforms, the developer might not know such requirements for all platform components, especially reacting the dependencies between the IoT part and the cloud part. For example, a developer might not understand the cloud storage performance required to the requirement of activating more sensors.

**Solutions:** To custom IoT cloud platform-specific analytics, we follow two different approaches: (I) bottom-up: common built-in metrics are structured and analyzed automatically, providing an overview over the platform's elasticity, and (ii)top-down: the platform developer can define custom, potentially domain-specific, metrics and analytics functions. Thus, we provide a complete end-to-end view over behavioural limits of the platform to enable the developer to redefine the platform, and improve its control strategies. Especially, it is crucial to defined an analytics function, which, based on supplied requirements, records encountered bounds on the monitored metrics not targeted by user requirements, in which the developer requirements were respected.

Elasticity Control Requirements: We need to enable elasticity for various parts of the IoT cloud platforms, such as sensors, gateways and cloud services, during the development and operation. This means that elasticity control mechanisms and tools must work across sub-platforms for design, test and operation purposes and must interface various protocols (e.g., REST, RMI, ssh + bash execution) used to change software components. Moreover, most developers would be interested in specifying abstract, high-level requirements (e.g., not focused on system-level metrics, controlling the software service as a whole).

**Solutions:** For sensors, developers could control the behaviour of sensors (e.g.,data reading frequency), to which gateway a sensor connects as well as the protocols between by gateways and sensors. At gateways, developers could control the number of sensor connections, the amount of data which is stored locally considering various constraints (e.g., the gateway has very limited computational power, memory and space). Moreover, we can add/remove various components for locally processing information, or change their sensitivity. For cloud services in an IoT cloud platform, we can support various known control mechanisms: (I)virtual infrastructure capabilities (e.g., add/remove virtual machines, network interfaces, disks), (ii) platform specific capabilities (e.g., start/stop web server,deploy/undeploy service in existing web server), or (iii) application-specific (e.g.,using API o_ered by cloud services developers). Capabilities. Each of these can be enforced separately or grouped into complex control processes. However, elasticity setup cannot be completely automated, and completely application-independent. In case developers need more advanced elasticity control, they can encapsulate them into their application specific control mechanisms (e.g., use aweb server deploy together with a new configuration, to result into other performance/cost characteristics). For controlling elasticity, we enable interaction based control to empower the developers with their control strategies,considering the evolution of the service at runtime.

## 5. EXPERIMENTS

### 5.1 Case Studies

Let us consider a scenario in which a predictive maintenance company would like to focus on predictive analytics for chillers in a city. The company wants to reuse/rent as much as possible IoT cloud infrastructures so that the company will focus on deploying its sensors, gateways, and cloud services. Both sensors, gateways and cloud services establish the company's IoT cloud platform. The IoT cloud platform includes gateways at the IoT part and cloud services at the data center. All of them are virtualized services, meaning that they can be deployed, configured and used on-the-way.

The predictive maintenance company will need features from the IoT cloud platform provider, which provides the right configuration of the IoT cloud platform for the predictive maintenance company. The IoT cloud platform can over features for a predictive maintenance company which monitors chillers and perform data analytics and maintenance tasks. In this case study, we will focus on the case the predictive maintenance company wants to buy services from an IoT cloud platform provider to create aconfiguration of its own elastic IoT cloud platform. Then the company develops and tests different sensors which connect to its elastic IoT cloud platform to have a complete system for gathering data to support data analytics1.

### 5.2 DEVELOPMENT AND DEPLOYMENT

First all of all, to make the (re)configuration of the IoT cloud platform, using our tool, the predictive maintenance

company can deploy two separate configurations: a configuration of the IoT sensors and gateways(A), and another of the cloud data center (B). This enables them to play with different sensors easily regarding data and topology, communication protocols, bursting workload, while the cloud services might be stable. For both configurations, IoT units and cloud services are provided from different providers from various repositories. The example of two topologies of sensors on two VMs, which allows the developer to manage single sensors and sensor topologies2 (and VM which host sensors insimulation scenarios). the deployment of an elastic IoT configuration platform named ElasticIoTPlatform at the data center to real cloud services and simulated gateways. With our techniques, such configurations (for sensors and for gateways/services) can be also programmed using Java code, enabling different ways to program and test IoT cloud platforms.

Having the entire IoT platform is provisioned, the company focuses on Monitoring(1 )). Before provisioning, platform developers must have in mind what monitoring data is relevant for the elasticity of the platform, and implement the necessary monitoring capabilities. A crucial factor in elastic platforms is that instances of units tend to appear/disappear dynamically at run-time as a result of scaling actions being enforced due to various elasticity requirements(e.g., platform performance, quality, cost).

Thus, the company wants to avoid monitoring information being lost due to scaling in/out of individual units, and also to have and overview over the overall behaviour of the platform units and not only individual unit instances. Thus, the platform developer must use our tool for deciding the contribution of a unit instance to the overall behaviour of the entire platform, or individual units, and structure monitoring information according to the architecture of the platform.

For example, the developer could decide that CPU usage of all unit instances must be averaged, and that the network data transfer. After having the platform deployed and monitored, the company focuses on the various Governance processes which must be enforced over the IoT sensors and gateways, arising from the company's different security, geo-political, performance objectives. For example, an abnormal event might be detected by the IoT platform, such as dangerous gas detected in a smart building. In such a case, for better analyzing the cause of the event, the frequency and data collected might need to be changed. For enabling such dynamic changes, we can invoke sensor and gateway capabilities through their APIs for changing data collection frequency, or execute a complex process for changing the security levels and protocols used to send data.

Leveraging these capabilities, we can enable processes for governing the gateways and sensors in different situations. Governance processes might change the frequency, size, and mechanisms in which sensor data is collected, processed and sent to the cloud data center. Thus, an Elasticity control mechanism is crucial for ensuring the performance and quality of the overall IoT platform, especially during and after the execution of governance processes, through elasticity. To enable elasticity control, the platform developers must design and develop elasticity capabilities for the individual platform units, wart, their type and purpose. Any capability that enables dynamic reconfiguration of any aspect or property of the platform unit's qualities as an elasticity capability, and must be designed and implemented in the platform units, and enforced at run-time.

For example, if a governance process increases the data collection frequency, the elasticity control mechanism should scale the platform to handle the load increase. One lesson learned is that from architectural design, development and operation, we need to decide if all of these complex services, gateways and sensors should be specified and deployed in a single software configuration or not. It is possible and it is hard to manage.

On the other hand, from an IoT cloud platform provider perspective, it is typical to provide a platform that includes gateways (at the edge) connecting to cloud services (in the data center) and let the customer to deploy possible sensors and configure these gateways and services to into the customer need.

### 5.3 ELASTICITY ANALYTICS AND CONTROL

After developing the ElasticIoTPlatform configuration, the developer can use our toolset for deploying and running it. At runtime, the developer is able to follow the behaviour of the application using our monitoring features, in order to redefine the elasticity and governance requirements and respectively policies. For such a complex use-case, which encompasses both IoT and cloud environments, there are two main control perspectives:
(I) controlling the services deployed in the cloud which manage data processing and storing, (ii) controls the IoT parts for addressing the governance policies.

In an emergency scenario, the entire ElasticIoTPlatform needs to react in order to localize or to better analyze the cause of the emergency. For this, further data needs to be collected, for avoiding errors and miss-predictions. a process described by the developer for addressing such case, in which the sensor push rate is increased (i.e., due to governance policy), and the cloud services are allowed to scale to higher cost levels. The latter is intended to address the issue of cost limit in the elasticity requirements, as normally the developer specifies a cloud service cost limit, for safety reasons.

In a day-to-day case, with an increasing workload the cloud service would employ more and more virtual resources up to the cost limit, while in the emergency scenario, the cloud

service can exceed the respective limit. From our experiences, we learned that, in our architectural designs, controlling elasticity of cloud software services should give powers to developers (e.g., controlling multiple software services at a Elasticity and governance process for the ElasticIoTPlatform configuration time, different software stacks, both system and application level metrics), while maintaining a simple mechanism of elasticity control specification. Moreover, this control of gateways or of sensors, should interface with a variety of tools (e.g., different cloud providers, using deferent protocols, different gateway vendor specific tools), for providing an end-to-end control of IoT cloud platforms.

### 5.4 DEPLOYMENT AND FAILURE

Let us consider some aspects related to the use of tools to evaluate IoT cloud platform deployment. We use our private DSG Open Stack, Stratus lab LAL public cloud We run our deployment engine in our private Open Stack with m1.medium VM (2 CPU and3,750 MB RAM) in the DSG cloud in order to test deployment issues for sensors and a configuration of an elastic IoT cloud platform ElasticIotPlatform.While we deploy ElasticIoTPlatform in our DSG cloud, we want to emulate several sensors by deploying them in both clouds where on each m1.small VM (1CPU, 2GB RAM)3, we deploy 30 sensors. We tested our studied configuration of an IoT cloud platform by deploying and activating from 100 to 350 sensors
when we use both clouds we deploy sensors equally in each cloud).an increasing and varying trend of deployment failure rates. We can seem that Flexiant has higher software failure rate by looking the deviation of failure percent of sensors and VMs, and VM failures are caused by the high number of concurrent requests on clouds.

## 6. RELATED WORK

Several challenges of IoT and cloud integration are discussed intensively .Many IoT platforms have been developed based on which different added services can be added. Our work is not about developing a particular IoT cloud platform, but focusing on techniques accelerating the development of such platforms. Although experiences have been shared, we have not seen similar experiences discussing rapid end-to-end development of elastic IoT cloud platforms. Several frameworks support the development of IoT. Industrial tools, such as Predix and Microsoft Azure IoT,also allow us to write IoT sensors and connect the sensors to cloud services. Butthey do not support elasticity controls. In our work, we do not focus on programming IoT sensors but recombine existing units and deploy them cross-issue spanning both IoT and clouds.

Such evaluations are useful for us to decide the infrastructure used for the cloud service part of the IoT PaaS. However, they have not focused on IoT clouds in general. The middleware, part of the Opinion platform , which provides functionality for dynamically adding/removing sensors to/from an IoT Platform spanning mobile networks and cloud infrastructures. We do not focus on particular platforms but we enable such functionality. There are some approaches on supporting simulation of IoT and IoT cloud systems. However, they are purely simulation systems, while we support configuration and testing of emulated sensors and gateways running in the cloud that interact with real-world cloud systems.

## 7 .CONCLUSIONS AND FUTURE WORK

In this paper, we described requirements, toolsets and engineering analytics for elastic IoT cloud platforms that simplify and accelerate the development of IoT cloud platforms, based on our development of the iCOMOT. Given the complexity of IoT cloud platform development requirements, it is hard any single, even powerful, toolset that will meet all the requirements. Therefore, we have to carry out appropriate engineering actions and also integrating different tools into our iCOMOT toolset. We show how utilizing such an integrated toolset we can simplify the development and and testing of IoT cloud platforms. Currently, we focus on building a common knowledge of components, topologies and artifacts for supporting testing and evaluation of uncertainties in elastic IoT cloud platforms, in particular, and cyber-physical systems, in general.

## REFERENCES

1. Crowd analytics archives. http://www.dfrc.ch/tag/crowd-analytics/
2. The controls galaxy, http://pacificcontrols.net/products/galaxy.html, Last access: 17 May 2016
3. U-test geo sports case study. http://www.u-test.eu/use-cases/#tab-1429727705-1-5
4. Akpinar, K., Hua, K.A., Li, K.: Thingstore: a platform for internet-of-things applicationdevelopment and deployment. In: Eliassen, F., Vitenberg, R. (eds.) Proceedingsof the 9th ACM International Conference on Distributed Event-BasedSystems, DEBS '15, Oslo, Norway, June 29 - July 3, 2015. pp. 162{173. ACM(2015)
5. Alamri, A., Ansari, W.S., Hassan, M.M., Hossain, M.S., Alelaiwi, A., Hossain, M.A.: A survey on sensor-cloud: Architecture, applications, and approaches. IJDSN2013 (2013), http://dx.doi.org/10.1155/2013/917923
6. Bonomi, F., Milito, R., Zhu, J., Addepalli, S.: Fog computing and its role in theinternet of things. In: Proceedings of the First Edition of the MCC Workshop onMobile Cloud Computing. pp.13{16. MCC '12, ACM, New York, NY, USA (2012)
7. Botta, A., de Donato, W., Persico, V., Pescape, A.: On the integration of cloudcomputing and internet of things. In: Future Internet of Things and Cloud (Fi-Cloud), 2014 International Conference on. pp. 23{30 (Aug 2014)