A Demo of IoT Healthcare Application Provisioning in Hybrid Cloud/Fog Environment

Ons Bibani‡§, Carla Mouradian†, Sami Yangui†, Roch H. Glitho†, Walid Gaaloul‡, Nejib Ben Hadj-Alouane§, Monique J. Morrow ¶, Paul A. Polakos*

†Concordia Institute for Information Systems Engineering. Concordia University, Montreal, QC, H3G 2W1, Canada

‡ SAMOVAR, Telecom SudParis, CNRS. Universite Paris-Saclay. 9, rue Charles Fourier. 91011 Evry Cedex, France

§UR OASIS, National Engineering School of Tunis. University of Tunis El Manar, Tunis, Tunisia

¶Cisco Systems, Zurich, Switzerland

*Cisco Systems, New York, NY, USA

***Abstract*—Fog computing brings cloud closer to end-users and data sources by enabling computation at the edge of the network. Low latency is the main benefit. IoT applications are often latency-sensitive. Such applications may be provisioned as component-based in a hybrid cloud/fog environment with components spanning cloud and fog. This will enable placing some of its components in the fog domain closer to the IoT devices, and consequently reduce the latency. However, provisioning applications in hybrid cloud/fog environment is still manual today. Existing PaaS do not support interacting with fog nodes, at the edge, for applications' components provisioning. This demo shows the key features of the hybrid Platform as-a-Service (PaaS) we have designed for IoT applications provisioning in cloud and fog environments. Three goals are assigned to the demo: (1) How applications can be designed and developed in such environments, (2) how the hybrid PaaS deploys the applications' components across cloud and fog nodes, and (3) how it executes and manages them using appropriate orchestration techniques.**

***Index Terms*—BPMN; Cloud computing; Fog computing; IoT**

I. Context, Motivations & Scope of the Demo

Internet of Things (IoT) enables the connection of various objects - such as Radio-Frequency IDentification (RFID) tags, sensors, cellphones over the Internet [1]. These objects interact with each other and cooperate with their neighbors through unique addressing schemes seeking to reach a common goal. This emerging infrastructure of objects could enable a plethora of new applications. The provisioning of these applications faces several challenges (e.g. efficiency in resource usage, elasticity) that cloud computing might aid in meeting.

In cloud setting, the Platform as-a-Service (PaaS) operates the virtualized physical resources provided by the Infrastructure as-a-Service (IaaS) in order to host and execute end-user applications. These applications are offered as Software-as-Service (SaaS) and may be located far from the end-users and/or data sources. This may cause unacceptable delays for latency-sensitive IoT applications such as firefighting and healthcare. Fog computing has been introduced recently. It extends the traditional cloud computing paradigm to the edges of the network by enabling computation and storage closer to the end-users and/or data sources when appropriate, to avoid excessive delays [2]. The edge of the network refers to the domain involving hosting nodes (e.g. cellphones, laptops) that are connected to the system within a distinct geographic area (e.g. local area network). With fog computing, applications' components could be either hosted and executed in the cloud PaaS or in the fog.

9781-5090-1445-3/16$31.00 © 2016 IEEE (CloudCom'16)

However, despite the fact that most of the existing PaaS solutions (e.g. Cloud Foundry, Microsoft Azure) enable provisioning IoT applications nowadays, all of them do not support provisioning these applications with components spanning cloud and fog. This is due to the absence of control, signaling, and data interfaces required to bind the cloud domain to the fog domains that could be part of such a hybrid environment. Furthermore, they do not support managing applications (e.g. executing, migrating, monitoring) with components distributed across PaaS and fog. Classical PaaSs are not able to achieve proper execution of the applications' flow with components belonging to an external domain than the main PaaS. In addition to that, monitoring and migrating applications components from cloud to fog (and vice versa) and/or from fog to another fog is very common in such hybrid environments for optimization purposes (e.g. [3], [4]). This complicates more handling the execution flow during the runtime since the applications' components locations may constantly change. It should be noted that all these operations are still performed manually in hybrid cloud/fog environments nowadays.

As part of our work, we designed a novel PaaS that enables provisioning IoT applications in hybrid cloud/fog environments. A high-level architecture of this PaaS was introduced in [5]. This demo shows the prototype that implements that PaaS architecture. It covers the whole applications' lifecycle. The prototype supports: (1) Developing and composing IoT application components, (2) deploying them, and (3) managing them including executing them by orchestrating the execution flow across the several involved components. Unlike classical PaaS, our prototype enables placing the applications' components in either cloud or fog when deploying them and is able to continue executing the whole applications during/after migration procedures. The prototype was validated using a set of realistic and latency-sensitive IoT applications such as *fire detection and fighting* and *robotic prescription dispensing and medication delivery* applications. The former is detailed

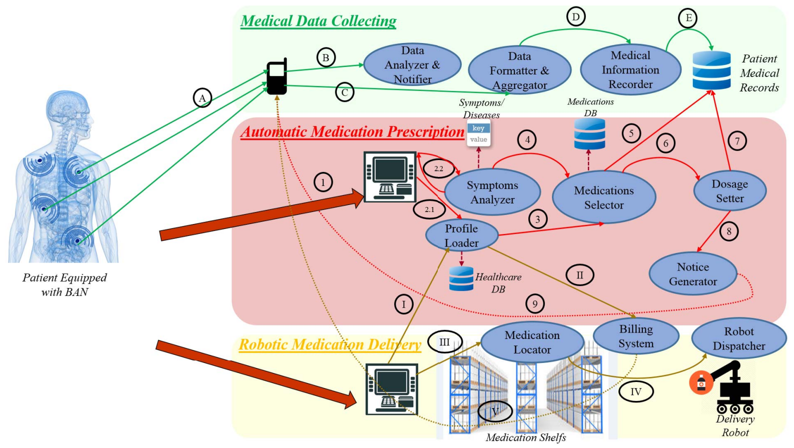

Fig. 1. Robotic prescription dispensing and medication delivery system

in [6] while the latter is used for this demo. The next section introduces the *robotic prescription dispensing and medication delivery* use case and discusses the prototype architecture. Section III is devoted to the demonstration overview. Section IV concludes the paper.

II. IoT Healthcare Use Case Description & Prototype Architecture

In this Section, we first introduce the IoT healthcare application use case that we use for the demo. Then, we discuss the architecture of the prototype that we implemented to provision such application in the hybrid cloud/fog environment.

A. Robotic Prescription Dispensing and Medication Delivery System

The use case we consider for the demo is depicted in Fig. 1. It implements an IoT healthcare application that consists of three subsystems:

- *Medical Data Collecting Subsystem*: for collecting and processing of the vital data related to a given patient. It is also responsible for formatting, analyzing, and storing these data.
- *Automatic Medication Prescription Subsystem*: for automatic medication selection based on patients' symptoms. This subsystem may interact with the *Medical Data Collecting Subsystem* in order to get the patients' medical record during the medications selection process. It generates a personalized notice for the patients.
- *Robotic Medication Delivery Subsystem*: for automatic medication delivery and payment based on the notice generated by the *Automatic Medication Prescription Subsystem*.

The detailed components, as well as, the execution flow of each subsystem are detailed in what follows:

1) Medical Data Collecting Subsystem: This system enables continuous monitoring and the processing of the vital data (e.g. blood pressure, heart rate, glucose level) related to a given patient. It handles vital information collected by a Body Area Network (BAN) deployed on the patient's body and routes it to the processing components. The routing is performed through connected close devices such as the patient's cellphone (Fig. 1, action A). The patient equipped with the BAN can perform his mobile, daily activities such as going to work and shopping. The monitored data is sent to the *Data Analyzer & Notifier* component (action B). This component parses the received data in order to detect a prospective heart attack. When the risk of heart attack is high, it immediately warns emergency services (e.g. calling 911) and/or a list of pre-stored contacts (e.g. family members) to warn them about the imminent heart attack. Simultaneously, the vital data is sent to the *Data Formatter & Aggregator* (action C). This component formats and adapts the received data according to a specific scheme (e.g. SQL, RDF format) for storage and indexation purposes. It may also encrypt the data for security and/or privacy purposes. The formatted data is routed to the *Medical Information Recorder* (action D). The latter stores the data for dynamic clinical assistance (action E). This data can be accessed by qualified professionals (e.g. the attending physician can carry out a remote diagnosis and update some instructions such as changing medication).

2) Automatic Medication Prescription Subsystem: This system enables selecting appropriate medications in automatic way by simply interacting with the patient. The patient starts by providing the system with his health insurance card and the list of his symptoms (Fig. 1, action 1). The system connects to the *Profile Loader* (action 2.1) which gets the patient's profile information (e.g. diabetes or not, surgical history, allergies to specific substances) from the health insurance database. This database belongs to the insurance company to which the patient is affiliated. These information are forwarded to the *Medication Selector* component (action 3). On the other hand, the provided list of symptoms is sent to the *Symptoms Analyzer* (action 2.2). This component implements an artificial intelligence search algorithm that aims at identifying the patient's disease. Specifically, it interacts with the patient when generating on the fly a set of questions based on the previous answers provided by the patient. Several iterations may be required in order to identify the disease. A connection to a key/value database that maps between the existing symptoms and the possible diseases is performed during this process. Once the disease is identified, the *Symptoms Analyzer* forwards it to the *Medications Selector* (action 4). This component processes a matchmaking between the identified disease and the medications information stored in the medications database. During this procedure, the *Medications Selector* may refine the list of the discovered medications using the information related to the patient profile (e.g. eliminating the medications that contain a substance to which the patient is allergic). It may also use his recent medical records (action 5) during the medication refinement process (i.e. to consider his recent glucose level when it comes to specific treatments such as water depletion). After that, the obtained list of medications is communicated to the *Dosage Setter* (action 6) in order to set the appropriate dosage for each one. A new connection to the patient medical records may be required (action 7) in order to get necessary information for optimal dosage such as the age of the patient and/or his weight. The medications list, as well as, the corresponding dosage, are sent to the *Notice Generator* (action 8). This component generates a personalized medications notice for the patient and sends it back to his cellphone (action 9).

3) Robotic Medication Delivery Subsystem: This system enables automatic medication delivery and payment. The patient provides this system with his health card and medications notice in order to get the prescribed medications. Information related to his health insurance profile (e.g. refund rate) are loaded from the health insurance database through the *Profile Loader* component (Fig. 1, action I) and forwarded to the *Billing System* component (action II). Simultaneously, the notice is sent to the *Medication Locator* component (action III). This component parses the notice and determines the location of each medication in the pharmacy (i.e. corridor, shelf, position). The list of the locations is then sent to the *Robot Dispatcher* component (action IV). The latter sends control actions to a delivery robot in order to collect the medications and give them to the patient.

B. Prototype Architecture

The implemented prototype for the demo is depicted in Fig. 2. It consists of three domains:

- The *IoT domain* that includes the BAN used for monitoring the patient's vital data. We use SHIMMER Platinum Development Kit as BAN. Shimmer is a lightweight body wearable wireless sensor platform designed for health-sensing applications. It allows capturing and communicating sensed data. We also use HTC 816 desire with Android V6.0 OS as cellphone to collect the data from the BAN, one STW chip card reader/writer to scan the information from the healhtcare insurance chip card and one LEGO EV3 Mindstorms robot as delivery device.

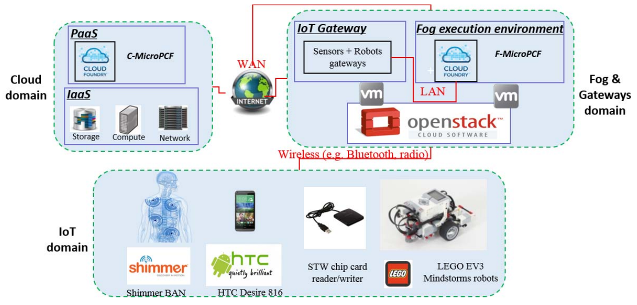

Fig. 2. The developed prototype architecture

- The *Fog and Gateways domain* is located close to the IoT domain and accessible via short range radio (i.e. Bluetooth). It is made up of the IoT gateway required for protocol and information model conversion [7] and of the fog execution environment required for hosting the applications' components located in the fog.
- The *Cloud domain* is further away and accessible via a wide area network. It includes the PaaS and the IaaS needed to execute the components located in the cloud.

It should be noted that the fog execution environment and the PaaS are based on an extended Cloud Foundry that enables the integration of the fog. Indeed, we extended MicroPCF, a lightweight distribution of Cloud Foundry, by adding novel components to support provisioning applications in the hybrid cloud/fog system. Examples of these components are Docker containers that host and execute applications' components, migration module to handle moving components from cloud to fog and vice versa. The detailed software architecture of the performed extension is presented in [5].

III. Demo Overview

This section describes the key features of our proposed demo. The demo uses the developed prototype and shows the whole provisioning process of the introduced healthcare use case in hybrid cloud/fog environment. The demo consists of three steps. Each step implements one phase of the application's lifecycle (i.e. development, deployment and management). Broadly speaking, our demo will show how one can (i) develop the IoT healthcare application in such environment, (ii) deploy it across cloud and fog according to a well-defined placement plan and (iii) execute it. Further details for each one of these steps are provided in what follows.

A. Development Step

For the development phase, we use the Eclipse IDE for code editing and application's components development. After that, each application's component is packaged in a Docker container with its specific requirements (e.g. scripts, libraries). In addition to the components source code, the developer has to provide a business process that describes the interactions between the components and the execution chain (the flow) of the whole application. We use Business Process Model and

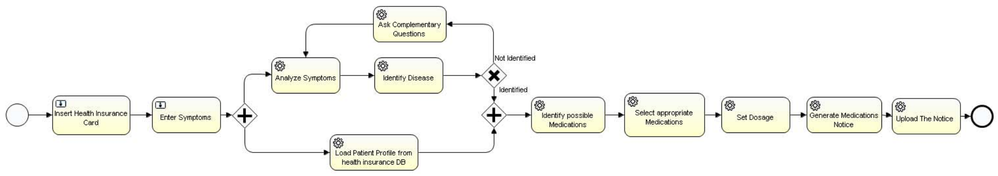
Insert Health Insurance Card
Enter Symptoms
Analyze Symptoms
Identify Disease
Ask Complementary Questions
Load Patient Profile from health insurance DB
Not Identified
Identified
Identify possible Medications
Select appropriate Medications
Set Dosage
Generate Medications Notice
Upload The Notice

Fig. 3. The BPMN orchestration workflow of the robotic prescription dispensing and medication delivery system

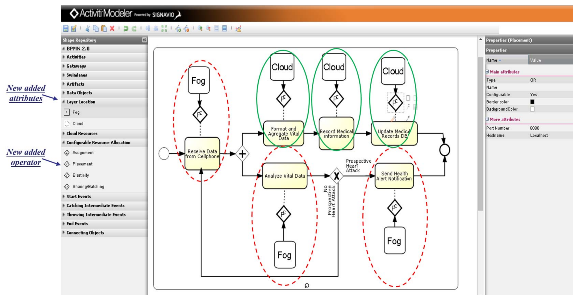
Activiti Modeler
New added attributes
New added operator
Fog
Cloud
Cloud
Cloud
Receive Data from Cellphone
Format and Aggregate Vital Data
Record Medical Information
Update Medical Records DB
Analyze Vital Data
Prospective Heart Attack
Send Health Alert Notification
Fog
Fog

Fig. 4. A screenshot of the extended Apache Activiti modeler interface

Notation (BPMN) to model such business process. BPMN is a standard for business processes modeling that provides a graphical notation for specifying business processes. As example, Fig. 3 depicts the BPMN process describing the *Automatic Medication Prescription Subsystem* of our IoT healthcare application. We use Apache Activiti modeler to draw the process. The activities of the process are the basis of the corresponding application orchestration workflow. They are designed as REST HTTP Web services. The multi-choice processes are handled through gateways. A gateway is an operator that evaluates the state of the business process and redirects the flow to the appropriate path based on its defined condition (e.g. exclusive gateway, parallel gateway). Back to the IoT healthcare application BPMN process, the *Load Patient Profile* and *Analyze symptoms* activities are processed in parallel after the *Enter Symptoms* activity. Finally, at the end of this step, the developer specifies explicitly the location of each application component (either in cloud or fog). To that end, we extended the Apache Activiti modeler in order to integrate a new "placement" operator and its related attributes (i.e. "cloud", "fog"). The extended elements, as well as, the placement decisions for the *Medical Data Collecting Subsystem*, are highlighted in Fig. 4. One placement possibility for this subsystem components consists on placing the latency-sensitive components (e.g. *Send Health Alert Notification*) in the fog close to the cellphone and the compute-intensive components (e.g. *Format & Aggregate Vital Data*) in the cloud.

B. Deployment Step

For the deployment phase, we use the Cloud Foundry Eclipse plugin to push the Docker containers within the application's components over the target MicroPCF instance. The Docker containers are then deployed over MicroPCF instances either in cloud or fog based on the developer placement decision for each component. For the demo, the MicroPCF instance part of the cloud domain (i.e. C-MicroPCF) will be deployed in the TSE Concordia lab in Montreal and accessible through the Internet while the IoT and fog & gateways domains, including the MicroPCF instance deployed in fog (i.e. F-MicroPCF), will be located locally in the demo room.

C. Management Step

For the management phase, we use Apache Activiti server as orchestration engine to execute the BPMN process associated to the application. We also use a migration engine that interacts with the Docker registry management framework and allows moving application's components within the Docker nodes from cloud to fog and vice versa.

IV. Conclusion

This demo aims at showing our performed prototype that implements provisioning of IoT healthcare application in hybrid cloud/fog environment. Its ultimate goal is to demonstrate the support of the whole application lifecycle.

Acknowledgment

This work is partially supported by CISCO systems through grant CG-589630.

References

[1] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A Survey," *Comput. Netw.*, vol. 54, no. 15, pp. 2787–2805, Oct. 2010.
[2] L. M. Vaquero and L. Rodero-Merino, "Finding Your Way in the Fog: Towards a Comprehensive Definition of Fog Computing," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 5, pp. 27–32, Oct. 2014.
[3] S. Agarwal, S. Yadav, and A. K. Yadav, "An Efficient Architecture and Algorithm for Resource P rovisioning in Fog Computing," *I.J. Information Engineering and Electronic Business*, vol. 1, pp. 48–61, 2016.
[4] R. Deng, R. Lu, C. Lai, and T. H. Luan, "Towards power consumption-delay tradeoff by workload allocation in cloud-fog computing," in *2015 IEEE International Conference on Communications (ICC)*, 2015, pp. 3909–3914.
[5] S. Yangui, P. Ravindran, O. Bibani, R. H. Glitho, N. B. Hadj-Alouane, M. J. Morrow, and P. A. Polakos, "A platform as-a-service for hybrid cloud/fog environments," in *IEEE International Symposium on Local and Metropolitan Area Networks, LANMAN 2016, Rome, Italy*, 2016, pp. 1–7.
[6] O. Bibani, S. Yangui, R. H. Glitho, W. Gaaloul, N. B. Hadj-Alouane, M. J. Morrow, and P. A. Polakos, "A demo of a PaaS for IoT applications provisioning in hybrid cloud/fog environment," in *IEEE International Symposium on Local and Metropolitan Area Networks, LANMAN 2016, Rome, Italy*, 2016, pp. 1–2.
[7] C. Mouradian, T. Saha, J. Sahoo, M. Abu-Lebdeh, R. H. Glitho, M. Morrow, and P. A. Polakos, "Network functions virtualization architecture for gateways for virtualized wireless sensor and actuator networks," *IEEE Network*, vol. 30, no. 3, pp. 72–80, 2016.