HAL
open science

BeC3: a Crowd-Centric Composition Testbed for the Internet of Things

Zahra Movahedi, Sylvain Cherrier, Yacine Ghamri-Doudane

To cite this version:

Zahra Movahedi, Sylvain Cherrier, Yacine Ghamri-Doudane. BeC3: a Crowd-Centric Composition Testbed for the Internet of Things. IEEE Consumer Communications and Networking Conference, Jan 2016, Las Vegas, United States. hal-01242960

HAL Id: hal-01242960

https://hal.science/hal-01242960v1

Submitted on 14 Dec 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

BeC3: a Crowd-Centric Composition Testbed for the Internet of Things

Zahra Movahedi, Sylvain Cherrier
University of Paris-Est
Paris, France
Email: name@univ-mlv.fr

Yacine M. Ghamri-Doudane
University of La Rochelle
La Rochelle, France
Email: yacine.ghamri@univ-lr.fr

***Abstract*—With the emergence of IoT devices, such as smartphones, temperature and light devices, etc., the ways of creating IoT applications has changed. IoT applications are often created and managed by a set of central points (orchestration) for different users. However, users may desire to create and manage their own applications based on their own logic in a decentralized way (choreography). Hence, in this paper, we demonstrate BeC3, a tool for creating and deploying Crowd-based applications using the choreography method. BeC3 is based on D-LiTE, a lightweight RESTful virtual machine designed for IoT devices. The users could then compose the D-LiTe-enabled devices using the BeC3 tool. BeC3 provides a simple and intuitive way to compose interaction between IoT components.**

I. Introduction

Since the number of IoT devices with simple functionality is increasing, a composition method to provide IoT applications with complex functionalities by composing simple functionaires is highly required. On one hand, some projects like FI-Ware[1], Octopus[2], and SENSEI[3] work on the interaction of IoT devices. Some other works in the domain of macroprogramming [1] are also interesting because they offer a simple and high-level solution to quickly create IoT applications. These works are important to build the IoT application, However, they are not destined to the simple users that are not familiar with the technical issues and require mostly the technical efforts for constructing IoT applications.

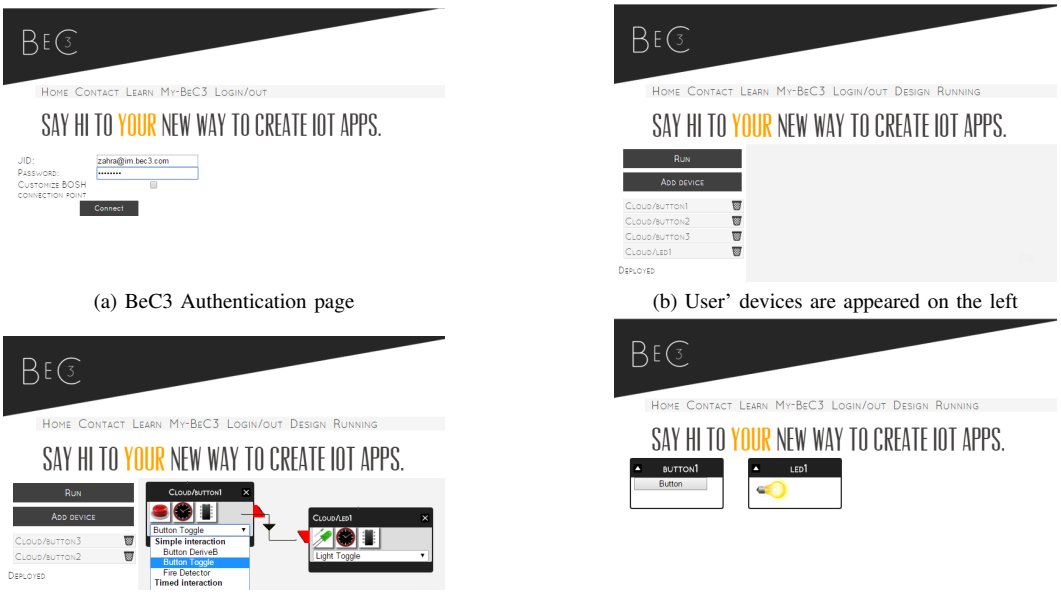
In BeC3, we believe that a user is both the developer and the architect of its own IoT applications. He may need frequently to reconfigure his applications according to his requirements. As a new approach, this paper demonstrates BeC3, a Crowd-centric tool that allows users to create and deploy IoT applications in a simple way.

A. BeC3 system overview

BeC3 tool has been developed based on two main elements: D-LITe and XMPP-REST. D-LITe is a lightweight RESTful virtual machine deployed on IoT device. It represents heterogeneous devices functionalities in an abstract way. It provides also a universal access to the functionalities of heterogeneous devices using XMPP-REST choreography approach. In D-LITe, a set of possible device' basic functionalities (called features) such as button, switch, timer, led, etc. are firstly defined. Each feature is driven by a set of potential small algorithms that specified the device functionality and control its usage. The behaviours are described using Transducers; A Transducer is an advanced form of Finite State Machines; i.e. each Thing is seen as a component with a current state, inputs, outputs, and transitions. Inputs and outputs are the message events exchanged by nodes through the network, and states are the nodes reaction to received messages. A transition links two states. A transition can be triggered by an input. States, transitions, and inputs describes the algorithm to be executed. Transducers add an output to the well-known Finite State Machines description. The output is generated by the Transition when triggered. The transducer representations used in D-LITe (and their specificities) are described with SALT [2], a simple description language that limits bandwidth and memory consumption.
D-LITe uses message exchanges between devices in order to compose devices interactions and to create IoT applications. To control the correctness of composition, typical messages exchanges (called Interaction Patterns) are defined. In this sense, two devices A and B could connect to each other if and only if the device A' output is matched with the device B' input using these Interaction Patterns. D-LITe allows an end-user to deploy a specific behaviour on each device. Each D-LITe enabled node contains a rules analyzer to execute the behaviour. D-LITe devices also have a messaging service to interact with each other using XMPP protocol, a standardized protocol for real time communication. This protocol offers instant messaging and presence management. Thus, the discovery of new nodes is dynamic and their integration in the global structure is easy. The XMPP-REST (an extension we have created for D-LITe, allowing to send REST commands through XMPP) handles behaviours on each device using GET, PUT, DELETE, and POST methods. GET RESTful method helps to discover existing features supported by the device, PUT method deploys a behaviour on a device, DELETE method removes an existing behaviour from the device, and POST method exchanges messages between two device behaviours. The implementation of REST approach within XMPP allows the use of the presence and chat mechanism offered by the instant messaging protocol, while this extension mimics the calls to a web service with the REST commands.

The Crowd-centric aspect allows a community-based design, granting a wide panel of modular and incremental interactions for a wide variety of components. Following this aspect, a set of different categories of users are involved in the BeC3 community to make the system functional. Indeed,

[1]http://www.fiware.org/
[2]http://www.octopus-project.eu/
[3]http://www.sensei-project.eu/



(a) BeC3 Authentication page

(b) User' devices are appeared on the left

(c) The user creates his IoT application

(d) The user deployed his created IoT application

Fig. 1: The BeC3 mashup tool: BeC3 provides an intuitive user Interface for the creation and deployment of IoT applications.

BeC3 relies on the 'participation inequality' [3] that describes the 01/09/90% rule. The 90% are simple users who want to create and deploy their own IoT applications with no required programming skills. The 9% create Behaviours that run on a specific object category. They are in fact FSTs that provide semantically "meaningful" usages for the 90%, and comply with Interaction Patterns to allow a maximum device interoperability. The last 1% takes care of implementing D-LITe on legacy devices such as sensors, phones and appliances. They may also punctually define new Interaction pattern.

B. BeC3 Demonstration

BeC3 front-end interface gives inexperienced users the ability to organise, interconnect and compose both state of the art web-services and IoT components to create interactive 2.0-like IoT applications. In this section, we provide a use-case scenario to present the BeC3 environment. We firstly start by the creation of a simple IoT application composing of a Button feature and a Led feature. The aim is to On/Off the light, toggling the Button. We then add some more features such as Timer, another more Buttons to enrich the created IoT application illustrating the dynamicity of the BeC3 system.

To use the BeC3 tool, each user should create an account with a unique ID and password. User' devices (here, the button, and the light) are firstly configured to connect to the XMPP server by giving the account of the owner, his password, and the node's name. We connect then to the BeC3 mashup tool with our account (Fig. 1a). When we authenticate on the BeC3 tool, the button and the light appear on the left side of the setting screen (Fig. 1b). To involve a device in our application, we drag icons from left side to the central panel, and choose a Behaviour from a proposed list of compatible ones. In our scenario, we selected the behaviour **Button toggle** for the Button device and the behaviour **Light Toggle** for the light device. Then, we link the button device to the light device, just by drawing arrows between them (Fig. 1c). Once it is finished, we try to send our choice to the devices. After checking the consistency of the assembly by the system, the BeC3 mashup tool sends messages to each device in order to describe the logic it has to follow (the Behaviour) and the observer's list of that node (arrows) using XMPP-REST (Fig.s 1d).

To enrich our application, we add a timer to the light device selecting the behaviour **toggle lighting timer**. We should also provide a duration for the timer when deploying the application. In this sense, after the deployment of the application, when we toggle the button, the light turn on and then turn off after the time duration. Furthermore, we add two other button devices with **Button toggle** behaviour, so that the light device can be turned on/off by each of these button devices. we can then add a timer to the light, etc.

II. Conclusion

We proposed and developed a crowd-centric tool named BeC3 for creating IoT applications. In the demonstration, we show that how a simple user can create and deploy complex IoT applications using the BeC3 tool. In the future, we aim to extend the BeC3 tool in order to support the multi-tenancy aspect for shared IoT devices.

References

[1] Luca Mottola and Gian Pietro Picco. Programming wireless sensor networks: Fundamental concepts and state of the art. *ACM Comput. Surv.*, 43(3):19:1–19:51, April 2011.

[2] S. Cherrier, Y.M. Ghamri-Doudane, S. Lohier, and G. Roussel. Salt: A simple application logic description using transducers for internet of things. In *Communications (ICC), 2013 IEEE International Conference on*, pages 3006–3011, June 2013.

[3] J. Nielsen. Participation inequality: lurkers vs. contributors in internet communities. *Jakob Nielsen's Alertbox*, 2006.