Integration of Heterogeneous Services and Things into Choreographies

Georgios Bouloukakis, Nikolaos Georgantas, Siddhartha Dutta, Valérie Issarny

Integration of Heterogeneous Services and Things into Choreographies

Georgios Bouloukakis, Nikolaos Georgantas, Siddhartha Dutta & Valérie Issarny*

MiMove Team, Inria Paris, France.
firstname.lastname@inria.fr

Abstract. Internet-of-Things (IoT) protocols are constantly increasing in the research and industrial landscape. However, the current standardization efforts limit the incorporation of Things as first-class entities into choreographies. To tackle this interoperability barrier, we propose and demonstrate the *eVolution Service Bus* (VSB), a middleware solution targeted to enable the interaction between Things-based and business-oriented services. Particularly, we demonstrate the incorporation of a service/Thing into the following choreographies: *i)* temperature sensors interacting with a business-oriented service, and *ii)* business-oriented services interacting with a route planner service.

Keywords: Middleware, Service Bus, Protocol Interoperability, Binding Component

1 Introduction

Service Oriented Architecture (SOA) allows software entities to interact in standard ways. Choreographies [2] enable large scale integration of such entities via SOA in a peer-to-peer fashion. Interactions are realized using well known protocols such as SOAP and REST; and each service exposes its functionalities (operations, messages, etc.) by relying on XML-based standards (WSDL/WADL). The existence of standards, facilitates the development of frameworks and the wrapping of systems for interoperability. However, with the advent of paradigms such as the *Internet of Things* [4], major tech industry actors have introduced their own APIs to support the deployment of devices (Things). The resulting lack of standards limits the incorporation of Things as first-class entities into choreographies. To remedy this, heterogeneous protocols employed by Things, such as CoAP [6] supporting *client-service* interactions, MQTT [1] based on the *publish-subscribe* interaction paradigm, SemiSpace[1]. offering a lightweight shared *tuple space*, or Websockets [5] based on streaming interactions, should be made interoperable.

* The work is supported by the collaborative research associate team ACHOR (inria.fr/en/associate-team/achor) and the EU-funded H2020 project CHOReVOLUTION (chorevolution.eu).

[1] http://www.semispace.org

In this paper, we introduce and demonstrate the *eVolution Service Bus* (VSB, proposed by the CHOReVOLUTION project[2]. Its objective is to seamlessly interconnect services and Things that employ heterogeneous interaction protocols at the middleware level, e.g., SOAP and REST for Web services; CoAP, MQTT and WebSockets for Things. This is based on runtime conversions between such protocols, with respect to their primitives and data type systems, while properly mapping between their semantics. This also includes mapping between the public interfaces of services/Things, regarding their operations and data, from the viewpoint of the middleware. More specifically, middleware-level operations and data are converted, while their business semantics remains transparent to the conversion.

VSB follows the well-known Enterprise Service Bus (ESB) paradigm [3]. In this paradigm, a common intermediate bus protocol is used to facilitate interconnection between multiple heterogeneous middleware protocols: instead of implementing all possible conversions between the protocols, we only need to implement the conversion of each protocol to the common bus protocol, thus considerably reducing the development effort. This conversion is done by a component associated to the service/Thing in question and its middleware, called a *Binding Component (BC)*, as it binds the service/Thing to the service bus.

We introduce a generic architecture for VSB, which relies on the *Generic Middleware (GM)* abstraction. GM identifies and supports the basic interaction styles found in most middleware protocols: *one-way*, *two-way asynchronous*, *two-way synchronous*, and *stream* interactions. It also distinguishes between the two roles participating in an interaction, such as: *sender/receiver*, *client/server* and *consumer/producer*. In VSB, we define the GM API primitives, which rely on two main actions: *i)* a `post` action for sending a piece of data; and *ii)* a `get` action for receiving a piece of data. We then implement the primitives of every concrete middleware protocol by extending the GM API in the *Protocol Pool*.

In the following section, we present our solution for synthesizing lightweight BCs. Then, we provide details about the VSB implementation followed by the way it is to be used by developers. Finally, we demonstrate the incorporation of services/Things to choreographies.

2 System Overview

To enable the incorporation of any service/Thing into a concrete choreography, we elaborate a generic architecture for BCs. Particularly, a *Generic BC* specifies all the aforementioned interaction styles using our GM API. We determine the supported interactions and the middleware concrete protocol of each service/Thing through a generic interface description, which we call GIDL (similar to WADL/WSDL, but general enough to support a variety of protocols). To synthesize a new BC and bridge two concrete protocols (the choreography protocol and the service's Thing's protocol), we use the implementations defined in the Protocol Pool.

[2] http://www.chorevolution.eu

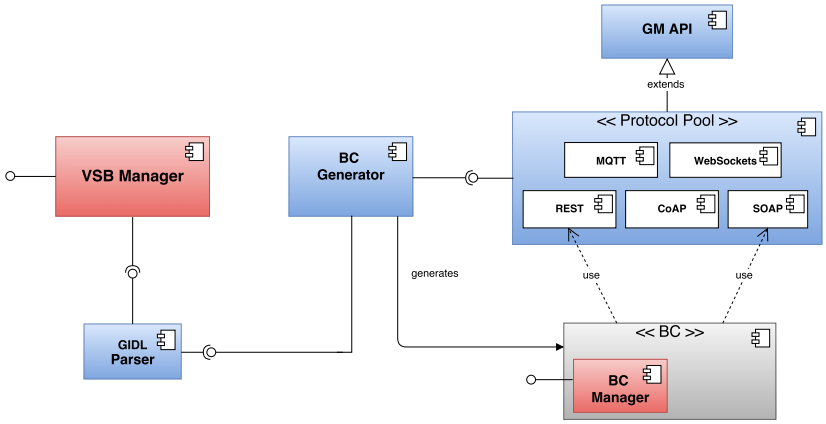

Fig. 1. VSB architecture

Implementation. VSB has been implemented using `Java 8` and the `Maven` software project management tool. A GIDL interface is in `JSON` format, inspired by the `Swagger`[3] specification. BCs are synthesized using the `JCodeModel` API (code generator). Regarding the *Protocol Pool*, REST has been implemented using the `Restlet` API, SOAP using the `JAX-WS` API, MQTT using the `Paho project` and CoAP using the `Californium` framework. The implementation is available as open source[4].

Use by the developer. We present the architecture of the VSB development environment in Fig. 1. VSB can be used with the following ways: *i)* an application developer incorporates a *new service/Thing* to the choreography *ii)* a middleware developer introduces a *new protocol*.

New service/Thing : A service/Thing is incorporated to the choreography through the following steps: *i)* the *VSB Manager* requires the service's/Thing's *GIDL* interface and the choreography protocol (e.g., SOAP); *ii)* if the service's/Thing's protocol (e.g., SemiSpace) is not included in the *Protocol Pool*, the VSB Manager cannot create the corresponding BC. In such case, the middleware developer must enrich the Protocol Pool with the service's/Thing's concrete protocol. *iii)* in case the protocol is supported (e.g., CoAP), the VSB Manager requires the corresponding BC from the *BC Generator*; *iv)* an artifact with the BC is synthesized as described above; *v)* through this artifact, the BC can be deployed, configured, started or stopped its execution using the BC manager.

New protocol : To add a new protocol (e.g., SemiSpace) in our framework, a middleware developer should follow the steps below: *i)* identify the protocol's primitives with respect to the GM API; *ii)* implement the above primitives; *iii)* implement the methods for the deployment (startup, shutdown, etc) of each component role (client, server, etc); and *iv)* bind the protocol to the *Protocol Pool*. To facilitate middleware developers introducing new protocols, we intend to develop an eclipse plugin.

[3] http://swagger.io/
[4] https://tuleap.ow2.org/plugins/git/chorevolution/evolution-service-bus

Demonstration. We consider two choreographies of services interacting using two different protocols: MQTT and SOAP. Regarding the 1st, developers desire to use a REST service to gather all the information published by temperature sensors (MQTT publishers). Sensor data should be published using the method `POST` (one-way interaction) to the resource named `sensors` through the URI: `/sensors/{id}/temperature`. We map the topic of MQTT publishers to the REST service resource and we demonstrate this communication using a synthesized BC. Concerning the 2nd choreography (SOAP protocol), developers wish to get information from a legacy *route planner* service, provided by Google. Thus, a SOAP request, `requestRoute (origin, dest)`, should be sent to the REST service which accepts requests to the URI: `/directions/{origin}/{dest}` using the `GET` method (two-way synchronous interaction). For this, we synthesize a BC that exposes a *WSDL* interface and is able to accept the requests sent. SOAP operations are mapped to the REST resources and XML data is converted to JSON for both the requests and responses. The above use cases are demonstrated in the following video: `https://youtu.be/UgfM3810RS8`

3 Conclusion

To allow the integration of both services and Things in choreographies, we introduce a lightweight and fully distributed ESB, which allows interconnection among smart devices and services in a peer to peer way. Different protocols can be introduced as VSB's common bus protocol with the same easiness as for integrating support for a new middleware protocol of a service/Thing; additionally, there is no need for relying on a full-fledged ESB platform. Using our approach, legacy services can be integrated into choreographies without the need for editing. VSB BCs are built and deployed as necessary; hence, no BC is needed when a service/Thing employs the same middleware protocol as the one used as the choreography protocol. We intend to extend the functionality of BCs, e.g., under certain aspects of IoT and mobile applications, to support dynamic changes in choreography scenarios.

References

1. Banks, A., Gupta, R.: Mqtt version 3.1. 1. OASIS Standard (2014)
2. Barker, A., Walton, C.D., Robertson, D.: Choreographing web services. IEEE Trans. on Services Computing 2, 152–166 (2009)
3. Chappell, D.A.: Enterprise Service Bus. O'Reilly Media (2004)
4. Guinard, D., Karnouskos, S., Trifa, V., Dober, B., Spiess, P., Savio, D.: Interacting with the SOA-based internet of things: Discovery, query, selection, and on-demand provisioning of web services. IEEE Trans. on Services Computing 3, 223–235 (2010)
5. Lubbers, P., Greco, F.: Html5 web sockets: A quantum leap in scalability for the web. SOA World Magazine (2010)
6. Shelby, Z., Hartke, K., Bormann, C.: IETF RFC 7252: The Constrained Application Protocol (CoAP) (June 2014), `http://www.rfc-editor.org/info/rfc7252`