Chapter 10
Interactions

Martin Bauer, Mathieu Boussard, and Stefan Meissner

As discussed in Sect. 8.2.2 and found in the literature, the functional view of a concrete architecture typically consists of three viewpoints: functional decomposition (viz. the logical structure), interfaces, and behaviour. Despite its significantly more abstract nature, we provide an analysis of these viewpoints for the IoT Reference Architecture in Sect. 8.2.2 and in Carrez et al. 2013: Annex C. However, only rudimentary interaction analysis is presented in the latter section, focusing mostly on technical use cases within a single FG.

Nevertheless, as can be appreciated by looking at already existing IoT systems, the operation of such systems generally involves sequences of FC interactions from all FGs. To help the reader better understand how common system-wide scenarios can be realised using the IoT ARM, and further apply this knowledge to their concrete architecture, this section provides the reader with an analysis of interactions between FCs across different FGs for some selected scenarios.

As explained earlier, the very nature of the IoT ARM is to cover all usage domains and architectures that can be derived from it – therefore it is not feasible to describe every possible FC interaction sequence for every possible scenario and architecture combination. Furthermore, instantiating a given scenario implies in most cases taking some clear Design Choices, before one can illustrate them in terms of FC interactions.

M. Bauer (✉)
NEC Laboratories Europe, Software & Services Research Division, NEC Europe Ltd.,
Kurfürsten-Anlage 36, 69115 Heidelberg, Germany
e-mail: Martin.Bauer@neclab.eu

M. Boussard
Alcatel-Lucent Bell Labs France, Route de Villejust, 91620 Nozay, France
e-mail: mathieu.boussard@alcatel-lucent.com

S. Meissner
University of Surrey, Stag Hill, GU2 7XH Guildford, UK
e-mail: s.meissner@surrey.ac.uk

A. Bassi et al. (eds.), *Enabling Things to Talk*, DOI 10.1007/978-3-642-40403-0_10,

However, in order to provide the reader at least with a general understanding of how such interactions can look like, we provide analyses for a few usage scenarios. The scenarios presented in the following sub-sections address some of the most representative system-wide general use cases, identifying relevant FCs and proposing an analysis of possible Design Choices when applicable.

The scenarios presented in this section are:

- Management-centred scenarios dealing with modification of the IoT system through
  - Configuration of the system when adding a device
  - Changing the device configuration
- Service-centred scenarios
  - Discovering relevant services using IoT Service Resolution and VE Resolution

Interworking of Service Choreography and IoT Services in the context of Complex Event Processing

10.1 Management-Centric Scenarios

This section presents the analysis of a "management-centric" scenario, namely the auto-configuration of an IoT system when adding a device or group of devices to the system. This scenario also encompasses the system-triggered configuration of such device(s) through the Management FG. Although the Device FG is out of the ARM (see Sect. 7.5.2), system designers have no choice but to consider this FG in the specification of their concrete systems. In particular, the interactions the FCs of the Device FG have with the entire system to make devices usable should be defined (we'll consider actual devices as distinct FCs here, but a device ensemble could be modelled likewise by a Device Group FC depending on chosen design choice).

In this section, we describe the interactions taking place across the Device, Management, Security, Communication, and IoT Service FGs for two management-centric scenarios, namely (i) what happens when a device is added to the IoT system and made available to its components, and (ii) what happens when a device configuration is changed within the system.

10.1.1 Configuration of the System When Adding a Device

From a general point of view, such addition can happen automatically, semi-automatically or in a manual fashion (which is a clear design choice). Common examples of these three different design choices are:

- For automatic joining, typically the handover between cells in a GSM network; Plug & Play solutions such as those supported by protocols like UPnP or Bonjour;
- For semi-automatic joining, the connection to a private network with a firewall, where a network administrator needs to manually grant access through inclusion of the requester's MAC address in a white list;
- For manual joining, any system where the complete compulsory information is manually inserted by an administrator (possibly including physical intervention).

The automated addition of devices is commonly addressed in concrete IoT systems through the usage of Plug&Play solutions (or a mix thereof). Extended to networked systems, Plug & Play conceptually covers addressing and more generally communication, resource description and discovery, registration and look-up as well as possibly secure and trusted access (see e.g. (Houyou et al. 2012)). Semi-automatic would e.g. imply equivalent discovery mechanisms but wait for approval of a human system manager to actually make the new device available to the rest of the system. Finally, some systems may not imply any automation at all – human system engineers perform static provisioning of necessary device information and trigger the addition of the device to the system when the physical deployment is performed. Regardless of the selected design choice, a number of actions need to take place, which are depicted below.

When considering an IoT system, the goal is to go from state A (system in initial state) to state B (system + new component in a state where this new component is actually potentially usable by the rest of the system components). In the following, we describe how the system might make this transition for two of the identified design choices (automatic or manual), as the semi-automatic case can be inferred as a mix of these two cases. The transition from state A to state B is caused by two types of triggers (Ta and Tb below):

- **Ta**: automatic design choice trigger (dynamic discovery/joining of the new device); the device is discoverable, e.g. actual (dynamic) appearance of the device in the range of the system (e.g. turning on the device, mobile device getting in range of the (e.g. radio) system);
- **Tb**: manual design choice trigger; the system is told to (statically) add a resource (or this specific resource); such a request can be issued within the system or by a human user.

Figure 10.1 below illustrates these two possibilities.

Triggers of type Ta) rely primarily on network-level mechanisms (e.g. joining network when requested from the incoming resource, or discovery when polled by the existing system communication gateway) that are specific to the concrete system implementation choices (e.g. Bluetooth discovery mechanisms), or to service-level mechanisms over a pre-existing network connectivity (e.g. UPnP over an IP-based local network), or a mix of both. From a concrete system perspective these mechanisms have to be supported both by the new resource and the system. Further aspects are discussed below.

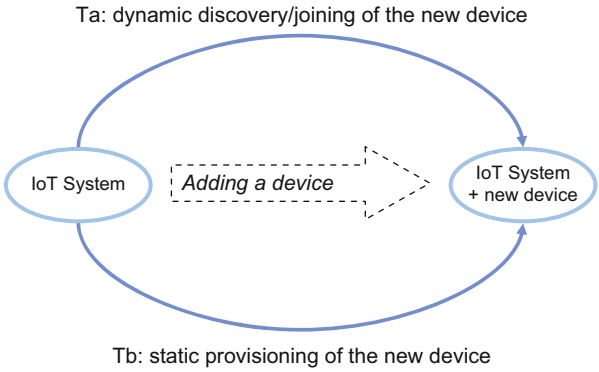

Fig. 10.1 Alternate paths to designing the addition a Device to an IoT System

Triggers of type Tb) are typically issued from a management function either upon explicit human (system manager) provisioning the configuration data for the new device (e.g. using a management console), or from another (non-management related) functionality in the system towards the management function (e.g. the IoT Service Resolution FC asking Management FG to add a new resource when it can't find one already available matching a given request)

From the Reference-Architecture point of view, the steps of the process when going from state A to state B shall include the following activities (represented in Fig. 10.2). Note that all the following steps should involve the necessary security measures for access control, namely authentication, authorization through respectively the Authentication and Authorization FCs of the Security FG:

- Update of Management FG components: in particular the Member FC's UpdateMember() interface should be called (see (Carrez et al. 2013); 7.4 Member)) – as the corresponding entry does not exist in the Member Database, it actually makes an "add" rather than an "update"). Other Management FCs can be impacted as well. The Configuration FC may retrieve and store the configuration of the new components, i.e. the resource and the collateral updates to existing components in the IoT Reference Architecture. The State FC may reflect the change of state of the system incurred by the addition/updates of these components. The Fault FC may have to correct related alarms, for instance if the former absence of the newly added devices incurred an alarm on the system. For instance in an IoT system controlling water level in a river with actuators offline due to maintenance, which raises an alarm in the Fault FC: as actuators go back online after maintenance, the system detects their re-appearance; the State FC is updated, and the Fault FC restarts regulation services as a consequence of the clearing of the alarm;
- As is the normal way to make use of a device through its associated IoT Resource, which itself is exposed through an IoT Service FC, the IoT Service FG needs to be updated, by creating a new IoT Service to represent this new IoT Resource (if necessary), and by updating the IoT Service Resolution FC (through its `insertService()` or `updateService()` interface);

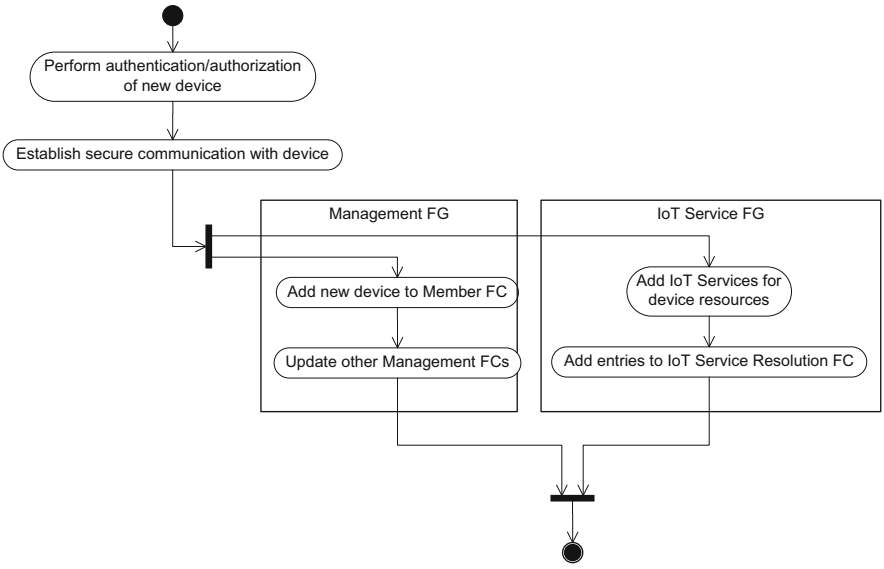

Fig .10.2 Adding a new device to the system – activity diagram

- If not already available (e.g. the IoT Service is newly created, or was not bound to the actual resource so far), the communication link between the IoT Service and the actual resource from the Device FG needs to be established, taking into account the specificities of the resource (e.g. intermittent availability) and desired communication patterns (cf Information View Section).

10.1.2 Changing the Device Configuration

In this section, we discuss how a device or a group of devices can be configured using different FCs of the IoT Reference Architecture. Such process involves the following steps depicted in Fig. 10.3:

- As a pre-requisite, the communication link from the Device FG to the various IoT Resources should be established, relying on Communication and Security functions;
- The request for a configuration change is issued by a human system manager through a management console, or by a FC. It results in a call to the Management:Member FC (step 1) through the `retrieveMember()` interface followed by a call to Management:Configuration FC (step 2) through the `setConfiguration()` interface. Naturally, such calls are subject to access control through the Security: Authentication and Security: Authorization FCs – not represented);

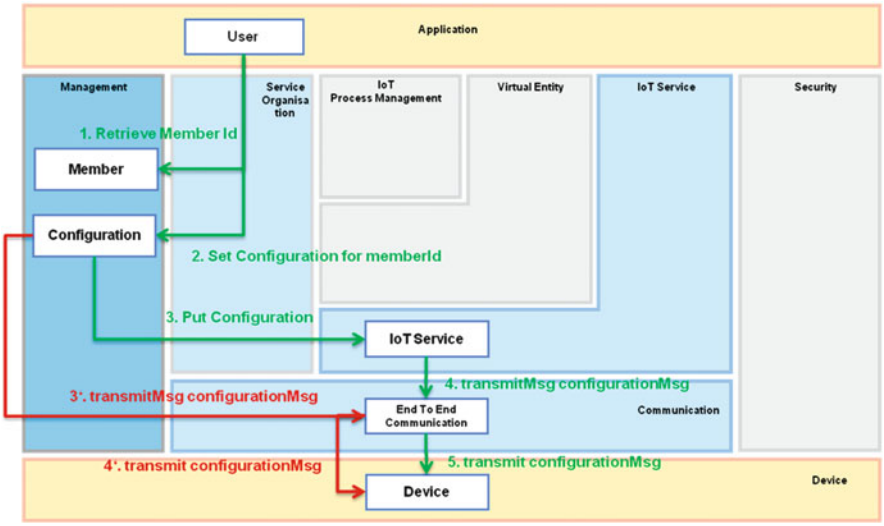

Fig. 10.3 Device configuration update interactions

- Depending on design choices made (e.g. whether the configuration of the associated resource is part of the related IoT Service or not), the actual configuration update on the device can be realized by either communicating directly with the device (e.g. sending a configuration message, steps 3′, 4′) or through the IoT Service associated with the resource (steps 3–5).

Please note that `prepareConfigurationMsg()` and `transmit()` methods are related respectively to the preparation of a usable configuration message to be transmitted by the End-to-End Communication FC, and to the actual reception of data on the Device itself, which are both out of scope of the IoT Reference Architecture, and therefore only shown here as an illustration. Underlying interactions with Security FCs and between End To End Communication FC and other FCs of the Communication FG are not shown (see ((Carrez et al. 2013); 5) on Communication FG).

10.2 Service-Centred Scenarios

10.2.1 Discovering Relevant Services Using IoT Service Resolution and VE Resolution

In existing, small-scale IoT scenarios, applications are often hard-coded or configured with respect to the sensors and actuators they are going to use. If we think of truly large-scale IoT scenarios, this is not going to be possible.

Applications should work in any environment, where the necessary infrastructure is available. This means that the necessary sensors and actuators first need to be found. The Functional Components responsible for this are the IoT Service Resolution and the VE Resolution. Due to the heterogeneity of the underlying hardware, and in order to make the functionality accessible in the whole IoT domain, it is desirable to provide a higher abstraction level than the hardware-level interface of the sensor. Therefore, the ARM offers a service abstraction level and a virtual entity abstraction level for the interaction with the IoT system. The IoT Service Resolution is the functional component responsible for discovering IoT Services based on a service description, which would typically include the service area; the VE Resolution is responsible for discovering the IoT Services associated to VEs, which can either provide information about the represented PEs or enable actuation on them.

In the following we look at a traffic scenario, but the interactions shown also apply to a large number of other scenarios. We have modelled the roads in form of road segments, where each road segment is a VE, and for each road segment, there is an associated sensor-based service that provides the road condition, e.g. whether the road there is dry, wet or icy. Figure 10.4 depicts the scenario.

To get to this scenario, the assumption is that either the IoT Services themselves or a management component, e.g. the Member FC, have registered each service together with their service area within the IoT Service Resolution. This is depicted as "Insert Service Description" steps (1)/ (1′) in Fig. 10.5. For the service areas only a few examples are shown. To simplify the discovery, road segments are modelled as Virtual Entities and associations between the road segments and the services are introduced. The respective IoT Services have service areas overlapping with the area of the road segment. The associations may be explicitly introduced by a service management component, e.g. the Member FC. (see Fig. 10.5 (2)) or they may be automatically discovered by the VE & IoT Service Monitoring component (see Fig. 10.5 (2′)), e.g. as the result of applying some rule on existing service descriptions from IoT Service Resolution and existing associations from Virtual Entity Resolution. The VE & IoT Service Monitoring component would then insert the newly created Association into the Virtual Entity Resolution component.

Now that the relevant information is available in the IoT Service Resolution and Virtual Entity Resolution, a car acting as a User that is driving along the road could then discover the services that provide information about the road conditions in the direction in which it is driving. Such a scenario is depicted in Fig. 10.6. The car would specify the geographic scope based on its current position and the driving directions, possibly taking map information into account. The scope is then used discover the associations between the upcoming Road Segments and the sensor services providing the respective road condition. Based on the service identifier, which is part of the associations, the service descriptions can be retrieved, so that the service can be accessed (Fig. 10.7).

The Application first discovers associations from the Virtual Entity Resolution looking for Virtual Entities of type road segment, with attribute road condition, within the geographic scope specifying an area that covers the road in the driving

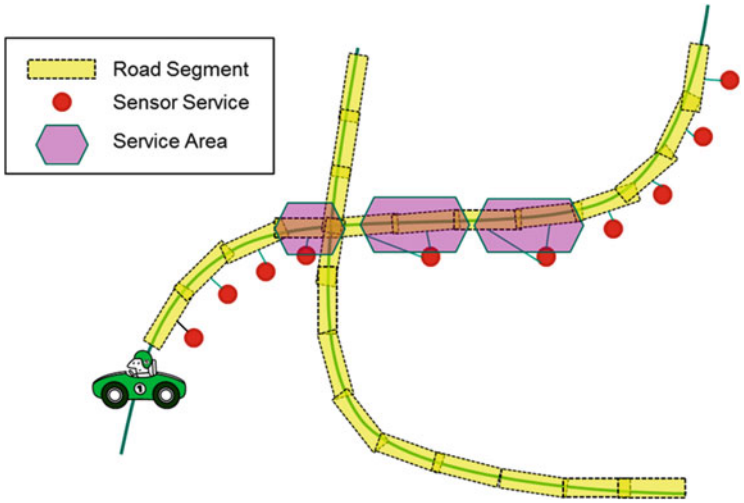

Fig. 10.4 Road condition scenario

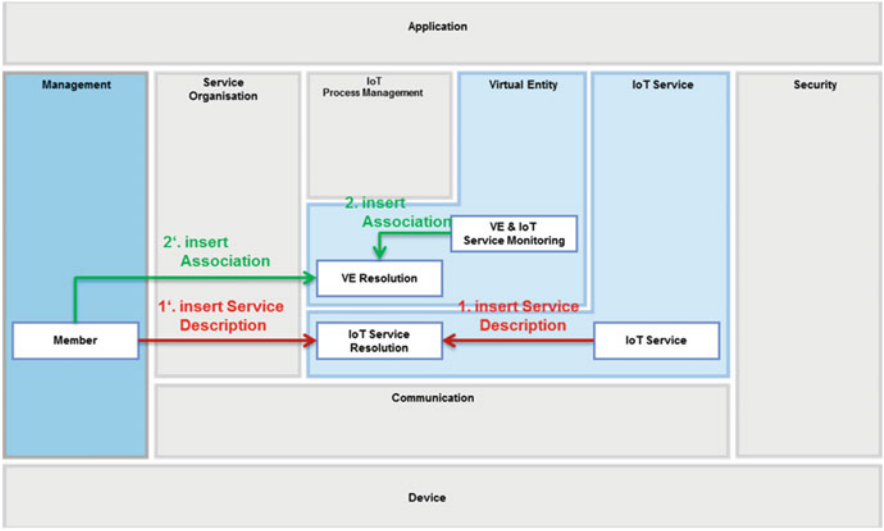

Fig. 10.5 Insertion of service descriptions and associations

direction (see Fig. 10.4 (1)). The returned associations contain the identifiers of the services that can provide the respective information. Based on these service identifiers, the service descriptions are looked up from the IoT Service Resolution (see Fig. 10.4 (2)). The returned service descriptions contain the information needed by the application to contact the respective IoT Services (see Fig. 10.4 (3)).

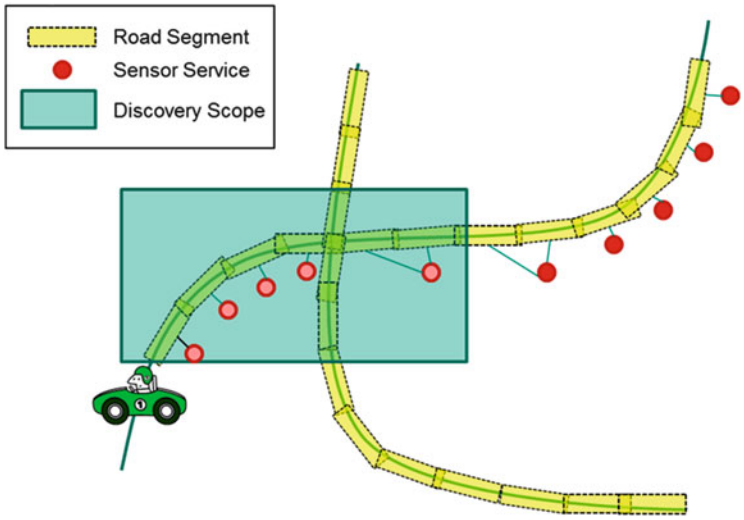

Fig. 10.6 Discovery of services providing information about the road conditions for the road segment in the direction the car is driving

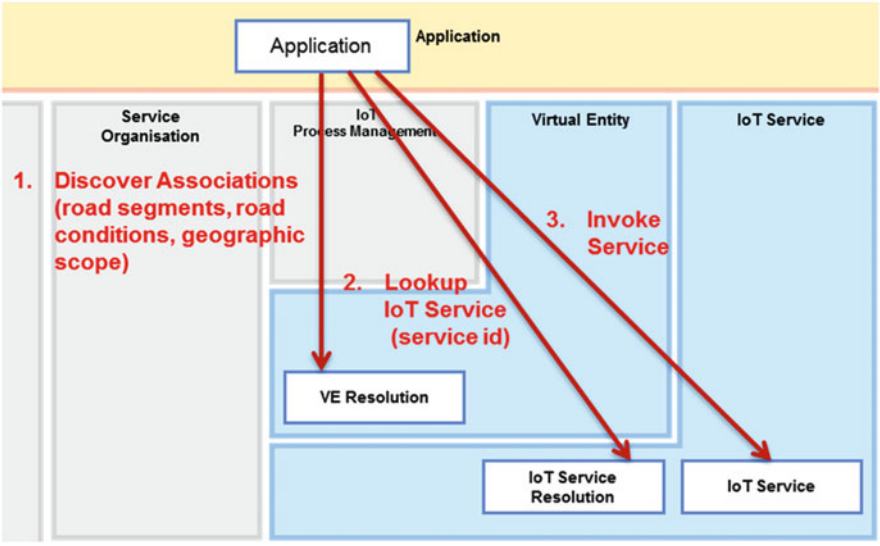

Fig. 10.7 Discovery and invocation of services providing the road conditions based on a geographic scope

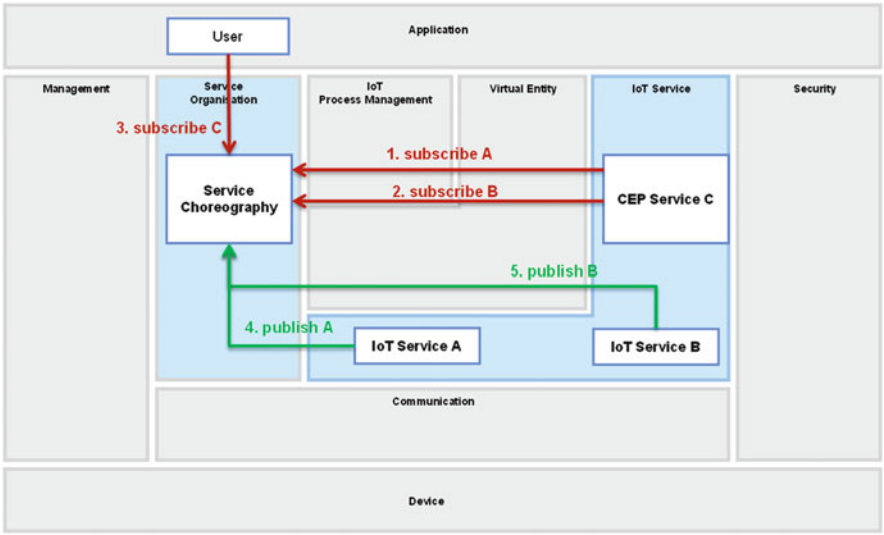

Fig. 10.8 Interactions `CEP Service C` subscribe

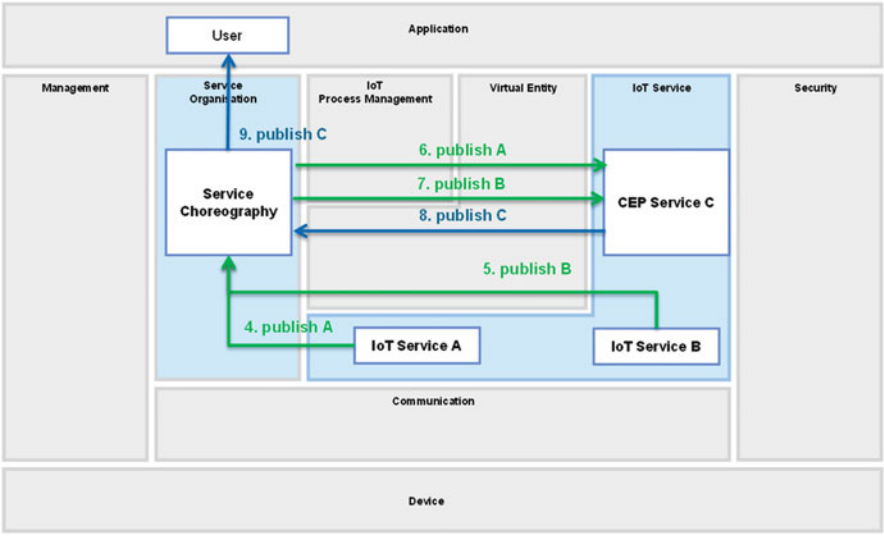

Fig. 10.9 Interactions `CEP Service C` publish

10.2.2 Managing Service Choreography

The FG Service Organisation contains the FC Service Choreography that supports Publish/Subscribe-functionality for IoT Services. In contrast to the IoT Service Resolution FC (Sect. 8.2.2.5) the Service Choreography FC contains a broker that can find suitable services for service requests given by potential service consumers. The service requests declaring an interest in certain IoT Service functionality are stored within the broker even if a suitable service is not available at the time the service request was given to the FC. As soon as a suitable service becomes available the broker receives the information the services publishes and forwards the information to the service consumer. On the other hand services can advertise their capabilities at the broker to await usage of potential service users. IoT Services can also publish information to the broker even if no service consumer is present.

In case multiple service consumers are interested in the information one particular service provides, the broker distributes the information to all subscribers (Sect. 8.2.3).

This Publish/Subscribe functionality allows using IoT services for CEP. In the scenario depicted in Fig. 10.8 the Design Choice has been made to provide CEP functionality as IoT Service, identified as CEP Service C. Such CEP services compute complex event based on simple events produced by other IoT Services (IoT Service A and B in Fig. 10.9). For this CEP Services need to subscribe to the IoT Services publishing the simple events (steps 1 and 2 in the figure below).

When the simple events are published to the Service Choreography FC (steps 4 and 5) the broker forwards the events to the `CEP Service C` (step 6 and 7). The `CEP Service C` is then able to process them to a complex event that is again published to the Service Choreography FC as illustrated in step 8 in Fig. 10.9

Since the user has subscribed to get notified if and when the complex event occurs (step 3 in Fig. 10.8) the Service Choreography FC publishes the event notification to the User as depicted in step 9 in Fig. 10.9.