Federation of IoT Systems and Service Function Chaining

Hidenori NAKAZATO[†a)], ***Fellow***

SUMMARY IoT systems tend to be silos which are built ground up for their particular applications. Fed4IoT is an EU-JP joint effort to federate IoT silos and make IoT devices and infrastructures shared among many IoT applications. This paper introduces the concept and the current architecture of Fed4IoT system. As a part of the Fed4IoT system, we are promoting to use service function chaining (SFC) to manage information to/from IoT devices within edge or in-network computing context. SFC provides a mechanism to reduce latency and distribute workload. Furthermore, Information Centric Networking (ICN) technology is adopted in addition to pub/sub communications, which is common among IoT systems, to implement SFC. The name-based forwarding in ICN gives us an opportunity to dynamically bind a function instance to a particular demand and consequently to meet the requirements in the demand. As an additional feature, the caching capability in ICN can be exploited to bring further reduction in latency and network load.

key words: *IoT, Federation, Service Function Chaining, ICN, Caching*

1. Introduction

IoT systems are expected to proliferate and to be applied to wide range of fields such as smart parking, traffic management, public transportation, vending machine management, smart home, and smart city. Most of the IoT systems are built ground up for their particular applications. Building an IoT system includes installation of IoT devices, connecting IoT devices to a communication network, and development of application software. Each IoT system is operated in isolation and interworking among systems is not easy. Say two smart parkings operated by different organizations cannot provide integrated view of open slots even though two parkings are located right next to each other. Each IoT system is a silo in other words.

There are already many initiatives to link IoT systems. However, it is not a simple matter to link IoT systems from different fields and provide services that span different application fields because these IoT systems use different interfaces and data models. Linking of IoT systems has not yet progressed beyond tests and demonstrations, and implementations with a business model and capable of sustained operation are still in the future.

Starting an IoT service incurs high initial cost due to the ground up IoT system construction. For example, a light pole is equipped with a motion sensor and there is a bus stop with a human counter right next to the light pole while a surveillance camera is installed targeting the area of the bus stop. Here we eliminate the motion sensor and human counter if we can share the surveillance camera. If we can share IoT devices, we can reduce the high initial cost by avoiding development of the infrastructure.

To deal with these two issues, federating IoT systems and avoiding the high initial costs, we are pursuing a joint Japan-Europe Research project under the Ministry of Internal Affairs and Communications, Strategic Information and Communications R&D Promotion Programme (SCOPE), called "Federating IoT and cloud infrastructures to provide scalable and interoperable Smart Cities applications, by introducing novel IoT virtualization technologies (Fed4IoT)." The project is a three-year project and started in 2018.

2. Fed4IoT System

2.1 Concept

The goals of Fed4IoT project are two-fold: making existing IoT system silos interoperable and lowering the initial cost to deploy IoT services. To fulfill the objectives, Fed4IoT project developed a few mechanisms to connect IoT systems adopting different IoT platforms and to share IoT devices. The mechanisms are IoT device virtualization, IoT platform virtualization, and data sharing infrastructure.

IoT devices are virtualized to be shared and reused. Here virtualization means preparing interface to IoT devices through software. The software provides the mechanism required for sharing and reuse. To share an IoT device among many users, access from the users need to be controlled. In particular, actuation from users must be properly regulated, or synchronized, to perform proper actuation. For example, one user tries to move a camera to the right while the other user is moving the camera to the left (Fig. 1). If we allow both users to control the same camera, we have a problem. Also, to promote reuse of IoT devices, preparing variety in

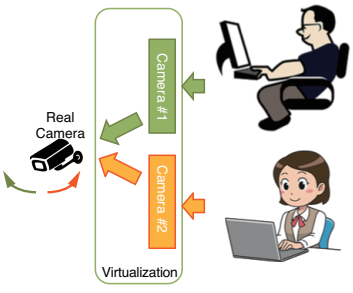


Fig. 1: Actuation Synchronization

[†]The author is with Waseda University, Shinjuku-ku, Tokyo, 169-0072 Japan.
a) E-mail: nakazato@waseda.jp

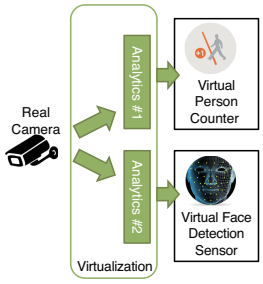


Fig. 2: Providing multple functions with virtual IoT devices

the functionality of the virtualized IoT devices can help. If a surveillance camera can provide person count and person finding capabilities (Fig. 2) in addition to simply providing the picture, the camera can be shared by more users.

To share IoT devices among many users, each user should be isolated from other users so that they can create programs without interference from the other users. This is like isolation in virtual machines in computing. To provide the isolated IoT service development environment, the concept of virtual silo (vSilo) is proposed. The IoT platforms contain IoT brokers where all information regarding IoT devices is stored and the information reflects the status of the IoT devices. Reading from the broker provides the current readings of IoT devices and writing to the broker controls IoT devices. Each vSilo is equipped with an IoT broker.

Then, virtualized shared IoT devices and IoT platforms of users must be able to communicate. The communication network should support multicast since one virtual IoT device may need to convey its data to multiple users. In addition, data format must be standardized within the network. The data is converted at the entrance and exit of the network to support different environments at both IoT device side and user side.

2.2 Architecture

Fig. 3 shows the system architecture of Fed4IoT. *Root Data Domain* at the left-hand side of the diagram houses existing IoT systems adopting heterogeneous IoT platforms. Real IoT devices belong to the IoT systems. Users are shown at the right-hand side of the diagram. Here the users develop their own IoT services and use them. What exists in between is the system developed by Fed4IoT project called *VirIoT*.

ThingVisor at the left in VirIoT virtualizes real IoT devices. ThingVisors act as the gateways to IoT devices in heterogeneous IoT systems. Each IoT system has its own idiosyncrasy in terms of access mechanism, information expression, and others. ThingVisors are responsible to resolve the idiosyncrasy and connect IoT devices to VirIoT. Typically, one ThingVisor is responsible in connecting one IoT device. It is the design decision in the ThingVisor, however. One ThingVisor may control more than one IoT device. Also, one ThingVisor may create more than one virtual IoT device as explained in Fig. 2. Each *vThing* in the diagram corresponds with one virtual IoT device.

ThingVisor acts as the arbitrator in actuation. If more than one user send actuation messages to one vThing or vThings, the ThingVisor in charge of the IoT device arbitrates the requests and controls the IoT device appropriately.

A *vSilo* at the right in VirIoT is an IoT service development and execution environment for particular IoT services. vSilo provides an isolated environment to run the IoT services. Each vSilo is equipped with an IoT broker which reflects the states of IoT devices and provides the access and control points for IoT devices. At this moment, VirIoT supports brokers for oneM2M [1], FIWARE [2], and NGSI-LD [3]. The user who want to have his/her own vSilo can create a vSilo with either oneM2M, FIWARE, or NGSI-LD flavor.

Data delivery platform in VirIoT is publish/subscribe (Pub/Sub) at this moment. Pub/Sub mechanism abstracts message destinations. The destinations of a published message are defined by the *topic* included in the message. The party intends to receive messages of a certain topic subscribes the topic at the rendezvous point of the Pub/Sub mechanism. Then, the party receives all the published messages with the specified topic. Pub/Sub provides multicast capability with abstract destination specification.

3. IoT and Service Function Chaining

Some applications of IoT systems require low latency. To support low latency *edge computing* is proposed. Also, a large number of IoT devices are expected to be connected even in one IoT system. By federating and connecting the IoT systems, and making one IoT application control a large number of devices, we foresee a large number of message exchanges between IoT devices and their server to handle the data. We conjecture that edge computing is not enough to handle the huge number of IoT devices and exploiting the *continuum of computing* spreading between edges and cloud data centers is necessary.

That is, the network itself needs to be a computing facility to process IoT data. Routers are equipped with computing facilities. The data from IoT devices is aggregated at the intermediate routers. The aggregation may be bundling data, statistical calculation, intermediate data generation among others. From this view of IoT systems, we are implementing ThingVisors as micro services deployed over the computing continuum, and let them cooperate. In other words, service function chaining is applied in making ThingVisors. For example, a chain of ThingVisors can create multiple vThings as shown in Fig. 4. The image taken by a camera is captured by a ThingVisor. The output makes a vThing where the captured image is accessible. The output of image capture ThingVisor is input to the human detection ThingVisor. The output of the human detection ThingVisor is propagated to both face detection and human counter ThingVisors, and so forth. By the chain of ThingVisors, three virtual IoT devices, or vThings: captured image, face image, and # customers become available without accessing the original camera for the three vThings separately.

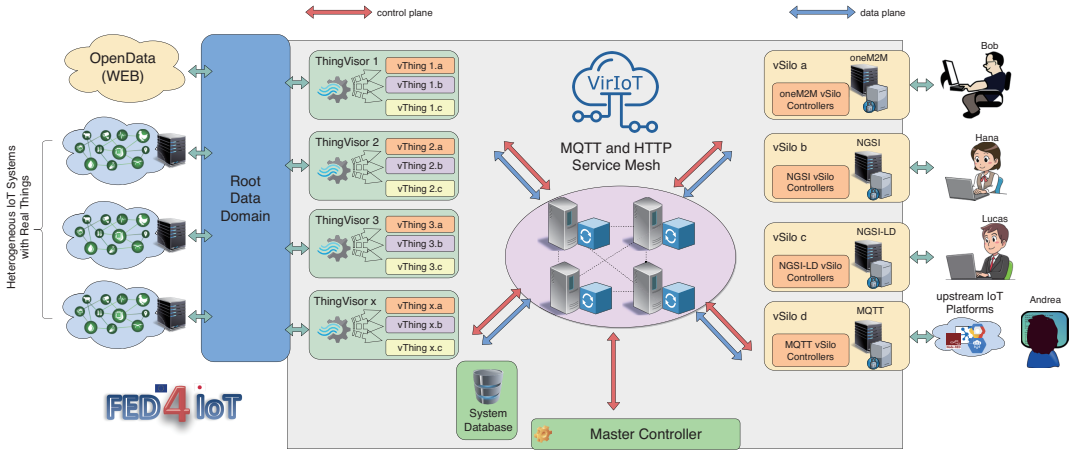


Fig. 3: Fed4IoT Architecture

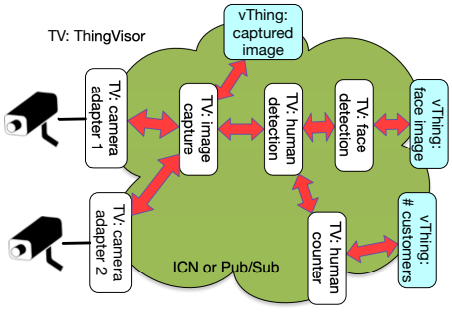


Fig. 4: Creation of vThings by SFC of ThingVisors

3.1 Networking for IoT Service Function Chaining

ThingVisors are created as micro services. Now, we need to connect them. As the network connecting ThingVisors we chose two types of networking: ICN and Pub/Sub.

Micro services are components to construct IoT services. ThingVisors are therefore potentially shared among many IoT services. For this reason, some of the ThingVisors will be more loaded than others. In order to avoid load concentration, some of the ThingVisors require multiple deployment and load need to be distributed among them. Abstract addressing in ICN and Pub/Sub helps distributing load among multiple ThingVisors. Topics are used to direct published messages to subscribers in Pub/Sub networking. Content names are used to direct interest packets to producers of the content. By using the abstract addresses: topics and content names, we have the flexibility to compose chains of service functions while distributing load.

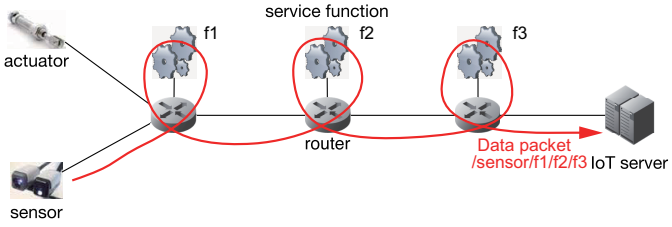
Up to now, Pub/Sub is the commonly used networking mechanism for IoT systems. Sensors with one type publish their readings with a topic. The IoT server who is responsible to the sensor type subscribes the topic. In this way, readings from all interested sensors can be collected by the IoT server.

For example, to distribute load using Pub/Sub, topic: "human detection" may be used to publish the output from the human detection ThingVisor in Fig. 4. The face detection ThingVisor and the human counter ThingVisor can receive the output of the human detection ThingVisor by subscribing to the topic "human detection." Since human detection is a computing intensive function, there may be several instances of the human detection ThingVisor and they may split the workload among the instances. The captured images transmitted by the image capture ThingVisor can be received by multiple instances of human detection ThingVisors. By configuring the human detection ThingVisors so that only one of the instances receive the image and apply its function, we can split the load among many instances. If all the human detection ThingVisors publish their result with the same topic "human detection," the face detection ThingVisor and the human counter ThingVisor can properly apply their functions to all the human detection information.

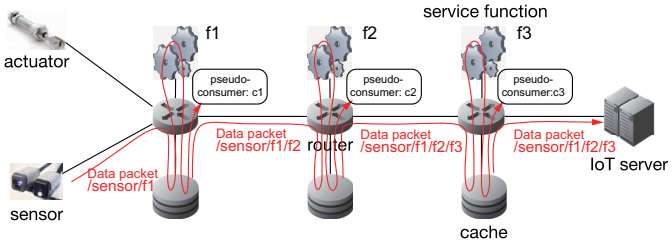
However, Pub/Sub is not handy for implementing SFC. Assume the output of ThingVisor: camera adapter 2 is used only for counting human. Then, human counter cannot simply subscribe to the output of human detector ThingVisor. Instead, the topic needs to be the one specifying the camera adapter 2 ThingVisor, an image capture ThingVisor, and a human detection ThingVisor, and also their application order. Then, there will be huge variety of topics for Pub/Sub.

Assume ICN is used instead of Pub/Sub. Then, we can specify the content name and a series of function names as the content name of interest packet to execute a chain of service functions. For example, to count the number of people in the picture taken by the camera connected to camera adapter 2 ThingVisor in Fig. 4, we may be able to express the chain by "TV:camera adapter 2/TV:image capture/TV:human detection/TV:human counter." Here the distribution of load is achievable by selecting an appropriate ThingVisors at the network level, i.e. routing.

We believe that both push type data delivery like Pub/Sub and pull type delivery like ICN are required for IoT networking. IoT devices with a tight power budget ask push type delivery. Sensors wake up to sense, push the measurements to the network, and immediately go to sleep mode to save energy. On the other hand, IoT services may not



(a) Without Cache



(b) With Cache

Fig. 5: SFC With and Without Cache

require data from IoT devices in the frequency determined by IoT device activation. IoT services may retrieve data less frequently. In this case, pull type data delivery fit more than the push type delivery. In this sense, both Pub/Sub and ICN should co-exist in an IoT network.

3.2 Caching in IoT Systems

One feature in ICN is content caching capability at routers. The content delivered in response to a request may be cached at any routers. Executing a series of service functions in SFC causes delay. Making use of cache in conjunction with service functions has potential to lower latency.

In ordinary SFC, a sequence of service function execution is triggered by reception of a request to execute a service function chain at the first service function in the chain. After finishing the execution of the first function, the next function is executed. Then, after the second function, the third function is triggered and so forth. A chain of functions is executed one after another. Fig. 5a shows an example. The IoT server requests `/sensor/f1/f2/f3` in a Interest packet. Here `/sensor/f1/f2/f3` means applying functions f1, f2, and f3 in that order to the sensor reading. The service functions f1, f2, and f3 are executed one after another.

In IoT systems, sensor readings are taken in a regular interval and actuators may be controlled in response to the sensor readings. In other words, service function chains are typically executed periodically. From this perspective, service functions do not need to wait until the arrival of the trigger from the previous service function in the chain, knowing the periodic triggering and having caching capability in hand. Instead, if the service functions are locally and periodically requested, the cache can be populated periodically. By repeating this local triggering at every service function in the chain, the links are connected eventually and make a chain of service functions as needed.

As shown in Fig. 5b, by enabling caching and crafting *pseudo-consumer* capability, each pseudo-consumer requests with its designated content and function names. For example, pseudo-consumer c1 requests `/sensor/f1`, pseudo-consumer f2 requests `/sensor/f1/f2`, and so on. Then, the cache at the router of c1 is populated with the result for the request `/sensor/f1`, and the cache at the router of c2 is populated with the result for the request `/sensor/f1/f2`, etc. The data taken from the sensor is processed by each function one after another and eventually delivered to the IoT server though the activation of the functions are not directly triggered by each other.

One issue to be solved in the *asynchronous* SFC formation is the end-to-end delay from the sensor to the IoT server. If the activation intervals of the pseudo-customers and their activation phases are not well configured, the end-to-end delay can get long and the sensor reading may become tardy by the time it is received by the IoT server. Unless fresh data from the sensor is required for a chain, however, this asynchronous configuration of SFC reduces response times and may be favorable for some applications.

How to configure the interval and phase of a pseudo-consumer while the ourput of the function is used by multiple service functions requires further study including the setting of the values and the mechanism to set those values.

4. Conclusions

Fed4IoT which aims to federate heterogeneous IoT systems and is an EU-JP joint research project is introduced in this paper. At the time of this writing, VirIoT, the federation mechanism of Fed4IoT, adopts Pub/Sub as its communication mechanism. We are extending the communication mechanism to include ICN to well support micro service style implementation of ThingVisors. As an additional plus, we showed that caching capability in ICN can be exploited to reduce response time of SFC executions.

ACKNOWLEDGMENT

The research leading to these results has been supported by the EU-JAPAN initiative by the EC Horizon 2020 Work Programme (2018-2020) Grant Agreement No.814918 and Ministry of Internal Affairs and Communications “Strategic Information and Communications R&D Promotion Programme (SCOPE)” Grant no. JPJ000595, “Federating IoT and cloud infrastructures to provide scalable and interoperable Smart Cities applications, by introducing novel IoT virtualization technologies (Fed4IoT).”

References

[1] oneM2M, “oneM2M web page.” [Online]. Available: https://onem2m.org, Last accessed on July 16, 2019.
[2] “FIWARE web page.” [Online]. Available: https://www.fiware.org.
[3] ETSI, “Context information management (CIM); NGSI-LD primer,” Recommendation ITU-T ETSI GR CIM 008 V1.1.1 (2020-03), ETSI, March 2020.