Fog Services Provider Architecture for IoT

Hoan Le, Nadjib Achir, Khaled Boussetta

▶ **To cite this version:**

Hoan Le, Nadjib Achir, Khaled Boussetta. Fog Services Provider Architecture for IoT. NoF 2020 - 11th International Conference on Network of the Future, Oct 2020, Bordeaux, France. pp.8-15, 10.1109/NoF50125.2020.9249177 . hal-03045499

HAL Id: hal-03045499
https://hal.science/hal-03045499

Submitted on 28 Jan 2021

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.

Fog Services Provider Architecture for IoT

Hoan Le
L2TI, Université Sorbonne Paris Nord,
Villetaneuse, France
hoan.le@univ-paris13.fr

Nadjib Achir
L2TI, Université Sorbonne Paris Nord,
Villetaneuse, France
Inria, Palaisau, France
nadjib.achir@univ-paris13.fr
nadjib.achir@inria.fr

Khaled Boussetta
L2TI, Université Sorbonne Paris Nord,
Villetaneuse, France
khaled.boussetta@univ-paris13.fr

***Abstract*—The prominence of Internet of Things (IoT) devices is characterized by a wide diversity of network access technologies, including Wi-Fi, Cellular, Lo-Ra, ZigBee, or Bluetooth. Dealing with such heterogeneity is still very challenging. Current cloud computing solutions can sustain a considerable amount of diverse data that are generated by geographically spread IoT devices. However, this task is quite challenging for time-sensitive services. Moreover, IoT heterogeneous devices are always managed by different application systems, and there are no unified storage and management solutions for information on IoT devices. In this paper, we propose a microservice architecture based on Fog computing paradigm named Fog services Provider (FSP) for IoT devices to perceive and flexibly manage heterogeneous devices in ubiquitous environments. A novel aspect of this service architecture is to build a general data model to describe and store data of IoT heterogeneous devices based on Semantic Web technologies. Besides, we also propose a data converting algorithm from relational databases to a unified format, and demonstrates procedures with effective management to register and access those devices into FSP architecture.**

***Index Terms*—Fog computing, device management, heterogeneity, distributed management, Internet of Things, Semantic Web, Interoperability**

I. Introduction

The Internet of Things (IoT) has evolved into a technological revolution providing a fully networked intelligent world, accelerating the 4th Industrial Revolution, where thousands or millions of things connected to the physical world. IoT is also a paradigm that takes into account the ubiquitous presence of a multitude of things or objects to collect and exchange data from a large number of heterogeneous devices such as Wi-Fi, Wired, 3G/4G, Lo-Ra, etc., via network access technologies. IoT has enabled the interconnection and intercommunication of billions of devices.

These *things* or *devices* have generated an unprecedented amount of enormous and heterogeneous data. Besides, data packets that are created from things can have various syntax, types, and formats, and the meaning of this data also varies from device to device. Traditional computing, like cloud computing, is an efficient means of processing and storing this type of data, but it still has to cope with a variety of problems. One of the biggest challenges is supporting time-sensitive applications. As a consequence, fog computing has recently emerged as a complement to cloud computing.

Fog services are being provided at the edge of the network and usually interoperate with cloud computing. As shown in Fig. 1. IoT things, fog and cloud nodes form a three-tier service delivery architecture. The difference between fog computing and other existing computing architectures is the proximity to the end-users. Indeed, one fundamental feature of fog computing is to support time-sensitive applications and its ability to keep computing resources at the edge of the network, and thus closer to the end-users. However, efficient management of IoT networks requires considering both the constraints of low power IoT devices and the deployment complexity of the underlying communication infrastructure.

The traditional paradigm of the IoT service model provides the software agent with raw sensor data collected by the devices. This raw data does not contain semantic annotation and requires a high manual overhead to implement practical applications. This diversity raises several interoperable issues. Because of the private approach of these service providers, the IoT domain has become a domain of vertical silos of different IoT applications without horizontal connectivity between them. This lack of interoperability with independent services currently poses a threat to the overall adoption and acceptance of the IoT domain, especially for applications that can profit from multiple devices. Thus, to achieve semantic interoperability in a heterogeneous IoT environment, the semantic annotation of raw sensing data using an ontology is a suitable approach. In general, all features of the semantic model are deployed centrally in the cloud. While the computing capabilities of the cloud are important, the response time for user requests is not always as fast as required by the applications. Besides, the use of semantic annotation algorithms to such amounts of data at the centralized layer leads to significant consumption of resources, which probably affects their performance. Consequently, a high-level fog computing architecture is required to cope with heterogeneous IoT devices generating large and diverse types of data. Concurrently, providing a general data model to support interoperability in IoT by using Semantic Web (SW) technologies.

In this paper, we propose a fog computing-based multi-technology services architecture for IoT to meet the requirements mentioned above. This architecture provides storage and management solution of IoT devices through the register and access services of IoT devices to the FSP. We focus on

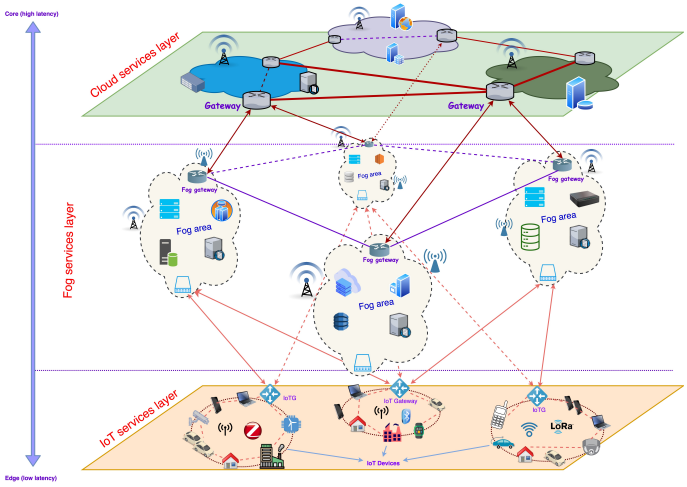

Fig. 1: The Internet of Things and Fog computing

the fog computing services architecture by controlling and managing the heterogeneous devices through the definition of their profiles and storing information based on semantic annotation service. The rest of the paper is organized as follows: Section II presents related works. Section III introduces our fog services provider (FSP) architecture for IoT and manages IoT devices through defining concepts and properties and relationships between data objects with the help of ontology and a converting algorithm from relational databases to a triplet. Section IV highlights its features thought real experimental Finally, section V, summarizes this paper and discusses some of our prospects.

II. Related work

In this section, we investigate some of the existing works on computing architecture for IoT, based on typical characteristics of fog computing paradigm. Bonomi *et al*. [6] proposed the first definition of fog computing for the Internet of Things. The authors point out some main features of fog computing to support IoT environments, such as low latency and location-awareness, geographical distribution, mobility support, and device heterogeneity. Rakman *et al*. [15] introduced a semantic-fog model with different functionalities such as data composing, or aggregation, modeling, linking, reasoning to model raw data with a conversion technique from processed data to annotate RDF data format semantically. They also propose an efficient offloading technique to reduce task execution time and energy consumption of the fog nodes. However, the relationship between data objects and mapping is not mentioned in semantic-fog architecture.

The project OpenIoT [19] has provided an innovative platform for IoT/cloud to i.) integrate IoT data and applications within cloud computing infrastructures, ii.) deploy secure access to semantically interoperable applications and iii.) handle mobile sensors associated QoS parameter. But, semantic processing functionalities such as data modeling, data aggregation, and data linking are performed centrally in the cloud. Semantic interoperability in Heterogeneous IoT infrastructure for Healthcare [12] proposed an IoT-SIM scheme to track and monitor patients through three significant components (User Interface – UI, Semantic Interoperability – SI and Cloud services – CS). Where SI directly interacts with UI by exchanging information meaningfully with shared vocabularies. They distribute semantic operations such as aggregation and modeling in a close area with IoT devices, while the composing process, linking, and actuation are performed in the cloud.

Chien *et al*. [7] proposed a fog computing architecture, with a distributed semantic reasoning among fog nodes. The main advantages are the reduction of traffic load, latency, and security. However, they do not indicate the reasoning relation between data objects. Verma *et al*. [22] proposed a model that uses advanced techniques and services, such as embedded data mining, distributed storage, and notification services at the edge of the network to monitor patient health remotely in a smart home. However, the semantic annotations of raw data generated by the sensor are not mentioned in their monitoring system.

Yi *et al*. [23] proposed a fog platform that includes components related to authentication and authorization, offloading management, location services, system monitoring, resource management, and VM scheduling. Their experimentation implemented a comparison between Fog and Cloud according to several parameters such as latency and bandwidth and showed some advantages of fog to cloud. However, their proposal is not focused on IoT domains. Adhatarao *et al*. [2] proposed a fog computing-based gateway to integrate the sensor network to the Internet. Their architecture takes full advantage of an instance of *Information-Centric Networking* to introduce a Fog gateway in the sensors network. Unfortunately, this Fog gateway neither includes the naming scheme for IoT devices in the sensors network nor provides an approach to managing IoT multi-network technology. Tuan Nguyen Gia *el al* [9] have proposed a fog computing architecture to support mobility feature, using a handover mechanism by deploying gateways to compare *Received Signal Strength Indicator* (RSSI) values with threshold values to estimate node position. However, their use case focuses on the sensor node movement which connects a set of gateways setting up with different topologies and placed at a fixed location.

Finally, an architecture for a smart health monitoring system based on fog computing proposed by Kharel *et al* [13]. The authors proposed a LoRaWAN gateway to manage IoT devices with different network access technologies such as Zigbee, Bluetooth, Wi-Fi, LoRa, 2G/3G, LTE, etc. However, the authors do not mention how to organize the data structure and to manage information on IoT connected devices.

Our work aims at designing a microservice architecture and facilities that manage IoT devices using different technologies. We also investigate interoperability between devices instead of data and the use of the Semantic Web to facilitate sensor discovery.

III. FSP - A Fog Services Provider architecture

A. Preliminaries

This section provides some basic notions that are necessary for the rest of the paper. *IoT devices*, *IoT things*, or *IoT objects* will be referred to as *Producers and Consumers*, while FSP is the main core and responsible for providing services to Producers and Consumers.

1) *Producer*: is a device designed to detect events or changes in its environment. It is used to generate data and send information to other objects. In some applications, a producer is also a sensor or an actuator or can be a thing that contains a number of sensors.

2) *Consumer*: is a software entity (microservices, API, user apps) which enables to analyze, process data generated from producers. The capability provided to the fog service clients is to use the fog provider's applications running on a cluster of federated FSP nodes managed by the provider. This type of service is similar to the cloud computing Software as a Service (SaaS) and implies that the end-device or smart thing accesses the fog node's applications through a thin client interface or a program interface.

3) *FSP (Fog Services Provider)* node: The FSP node is the core component of the fog computing architecture. It is an entity which provides services, components, modules for producers, consumers who can communicate with FSP. It includes both hardware components (e.g. gateways, switches, routers, servers, etc) and software components or virtual components (e.g. virtualized switches, virtual machines, cloudlets, etc.). It can be tightly interacted at a high level and low level with the smart end-devices or access networks, and provide computing resources to these devices. Additionally, FSP nodes provide some form of data management and communication services between the network edge layer where end-devices reside. To deploy a given fog computing capability, FSP nodes operate in a centralized or decentralized manner and can be configured as stand-alone fog nodes that communicate among them to deliver the service or can be federated to form clusters that provide horizontal scalability over disperse geolocations, through mirroring or extension mechanisms [11].

B. Overview architecture of Fog computing

Fog computing is defined as a distributed computing paradigm. It provides and manages more simply computing, storing, and networking services between data centres and producers/consumers. Fig. 1 presents a three-layer architecture for fog computing.

1) *IoT services layer*: This is the bottom-most layer or the closest layer to the end-user and physical environment. It encompasses all various IoT devices such as sensors, mobile phones, smart vehicles, smartwatches, cameras, etc. They can use several different types of connectivity like Bluetooth, ZigBee, Wi-Fi, LoRa, MQTT, 3G/4G, LTE, etc. Raw data or events collecting from producers do not contain any semantic annotation and demand extensive manual effort by using practical applications.

2) *Fog services layer*: The intermediate layer, also known as the fog computing layer. This layer is located on the edge of the network. It comprises of a distributed a large number of fog nodes which including routers, gateways, switches, access points, base station, micro datacentres and fog servers, etc. Producers act at IoT services layer always equip low-processing capability and limited storage. Fog services layer provides a flexible and easy way to analysis and process raw data at the edge of the network instead of the cloud. Moreover, this layer has the capability to model processed data to make semantic contexts and provide interoperability at data annotation level through concepts, attributes and the definition of relationships between data objects. These modeled data then are mapped into a triplestore (a triple is a data entity composed of *subject-predicate-object*) or RDF (resource description framework) store by taking advantage of a web ontology language (OWL, a standard of W3C [1])

3) *Cloud services layer*: The cloud computing layer is the top layer in this architecture, this layer consists of multiple high-performance servers, datacentres and network systems which are capable of processing and storing an enormous amount of data and provides different applications and services.

C. Services catalogue in FSP

The FSP network architecture is designed in a hierarchy of levels. This allows processing, networking and storage at each level according to the topology and services running on the FSP. IoT will be a major service-oriented technology. Therefore, it must be easy to organize all levels of the Fog Hierarchy through many different classes of stakeholders. The advantage is that the FSP architecture is easy to create, modify and maintain.

To illustrate several published literature works, in Table. I we have separated and selected criteria for the formal definition of fog computing and related works, including *Fog Computing Architecture* (FCA): provides features for interpreting issues such as low latency and location, mobility support, high geographic distribution, a large network of sensors and actuators, heterogeneous devices; *Internet of Things* (IoT): focuses on features that must take into account the handling of IoT devices; *Programming Model* (PM): enables the deployment of the optimal solution for IoT applications, such as real-time responsive applications or the ability to use resources at the edge of the network; *Resource Provisioning Management* (RPM): provides services close to the data sources or gateways to reduce delays and enable better leveraging of available compute, storage and network resources in fog computing; *Dynamic Application Topology* (DAT): enables the deployment and interconnection of distributed services on any selected execution environment and can take into account the dynamic restructuring of the system topology and adapt to the underlying topology at run-time; *Security and Privacy* (S&P): provides the possibility of authentication and authorization at different levels to ensure privacy at the edge of the network; *Reliability* (REL): takes full advantage of fog computing

TABLE I: Comparison of the related fog computing architecture

Works	FCA	IoT	PM	RPM	DAT	S&P	REL	EEf	SDA
Bonomi et al. [6], [5]	✓	✓	x	x	x	x	x	x	x
Stojmenovic et al. [20]	✓	x	x	x	x	x	✓	x	x
Faruque et al. [8]	✓	✓	x	x	x	x	x	✓	x
Saures et al. [16]	✓	✓	✓	✓	x	x	x	x	x
Skarlat et al. [18]	✓	✓	x	✓	x	x	x	x	x
Singh et al. [17]	x	x	x	✓	x	x	x	x	x
Zhan et al. [25]	x	x	x	✓	x	x	x	x	x
Okafor et al. [14]	✓	✓	x	x	✓	x	x	x	x
Shanhe Yi. [24]	✓	x	x	x	x	✓	x	x	x
Rakman et al [15]	✓	✓	x	✓	x	x	x	✓	✓
FSP Platform	✓	✓	✓	✓	✓	✓	✓	✓	✓

architecture to achieve accurate results by minimizing overall delay and with high reliability by using edge services close to the endpoints; *Energy Efficiency* (EFf): provides the flexibility, interoperability, connectivity, privacy and real-time service required for energy efficiency management; *Semantic Data Annotation* (SDA): is the ability to map meaning to raw data and obtain knowledgeable information. Besides, to ensure semantic interoperability in heterogeneous IoT environments, semantic annotation of sensory raw data using ontology is an appropriate method.

We propose a four-layer micro-services architecture based on fog computing and the two underlying system services with service providing capabilities for IoT environments to meet the above criteria. These include the IoT producer layer, the FSP infrastructure, the FSP management services, the FSP support services, the security/private sphere management and the analysis and data management. Based on the advantages of a micro-services model such as *self-discovery*, *loose coupling* and *high cohesion*, this architecture is proposed to integrate services that manage a kind of multi-network technology of heterogeneous IoT producers and components in the FSP and are ubiquitously located in Fog areas. Fig. 2 shows a simplified services stack for the fog architecture in our proposal.

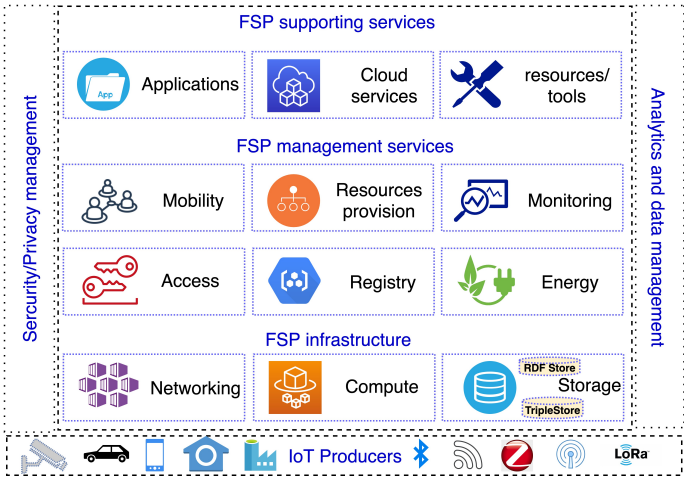

Fig. 2: Microservices architecture of Fog computing

1) *Access services* are responsible for handling the communication from IoT producers and consumers to FSPs via their native protocols. The two most important services are *producer service* and *consumer service*. The *producer service* plays a role as connectors that interact with the IoT producers. It can simultaneously serve one or more producers, including sensors, actuators, etc. In order to ensure semantic interoperability, this service provides a semantic methodology that focuses on interoperable IoT provisioning by converting the raw data produced and transmitted by the IoT producers into a common data structure called TripleStore or RDF storage and sending this converted data to the storage services at the FSP infrastructure layer and to other micro-services at other layers of the FSP. The *consumer services* are the means of extracting, processing/converting event/read data from FSP and sending it to an endpoint or process of the end-users choice. Besides, these services provide APIs or SDKs that can be used to build applications by assembling triggers, built-in functions, and custom functions for end-user decision making on specific services.

2) *Registry service*: The registry can is referred to any platform which may use for service discovery. The objective of the registry is to enable microservices to discover and communicate with each other. When each microservice starts up, it registers itself with the registry, and the registry continues checking its availability periodically via a specified health check endpoint. For example, when one the consumer microservice needs to connect to another one is the storage microservice to require data information (e.g. temperature or humidity), it connects to the *Registry* to retrieve the available hostname and port number of the storage microservice and then invokes the storage microservice to communicate.

3) *Monitoring service*: undertakes to analyze the monitoring data and dismiss events when previously specified QoS thresholds. This microservice can provide useful information such as work load, usage, energy to help lots of decision making and pricing. We highlight this component in fog computing architecture since it provides crucial information for other components.

4) *Resource provisioning service*: provides functions for orchestration, provision, and monitoring resources in the associated FSPs. It includes microservices such as *Resource computing service*, *Service placement*, *Resource reallocating*. To create an allocating plan to execute received task requests according to a specified resource allocating approach (selected

FSP to use) performs by *Resource computing service*. *Service placement* is responsible for controlling task requests by deploying services and depend on the *Resource computing service*. *Resource reallocating* enable to calculate a resource allocating plan according to the events (producer join/leave, producer failure).

5) *Mobility support service*: The handoff mechanism allows for continuous connectivity of data sources if network access is temporarily interrupted. Managing the multi-network for access by a large number of heterogeneous IoT producers is a key challenge in the cloud. Distributing fog resources at the edge of the network, where proximity to the IoT producer layer is a viable way to support mobile environments Furthermore, these IoT producers can change locations based on different types of communication and connections. It is very important to keep a location list of the IoT producers to be able to offer services such as motion tracking, location information sharing, etc. in the meantime. The mobility service in our FSP node handles the network locations with geographical information of producers and consumers and adopts the mobility management model provided by the FSP architecture through semantic data organization.

6) *Energy efficiency service*: moving computing, control, data storage and processing to the cloud face increasing constraints, such as reduced latency, high mobility, high scalability and real-time execution. Addressing the computing and intelligent networking requirements for next-generation network technologies (e.g. 5G). Fog nodes have deployed distribution services closer to end-users to solve the problems of energy consumption in the IoT environment by introducing a dynamic speed scaling mechanism at the edge of the network [21]. Therefore, an energy efficiency model is needed to manage and maintain the computation for IoT producers.

D. A data model for supporting interoperability in IoT

Semantic technologies enable computers to understand the meaning of data and to process data correctly and draw conclusions. Therefore, the application of semantic web technologies to IoT is becoming a crucial prerequisite for the integration of data and the future development of intelligent IoT applications and services in a variety of areas. IoT producers produce an enormous amount of raw data that is processed and made smarter by transforming expert information from raw data into semantic annotations using ontology. A typical producer service of the FSP architecture is the producer service, which links low-level raw data information with knowledge application services by enabling interoperability at the data modelling level to facilitate the heterogeneity between IoT producers and the FSP and provide APIs for interaction with different types of devices.

Fig. 3 shows a data model for heterogeneous producers. The low-processing components are mainly categorized into three major elements: IP-connected and non-IP producers, connection technology and semantic annotation. Typically, IP-connected and non-IP producers represent the lowest level and are composed of a set of very limited resources whose main

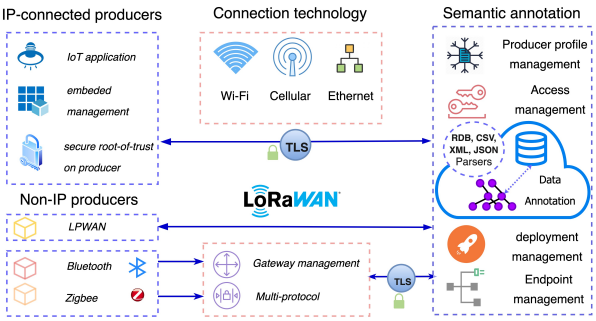

Fig. 3: IoT producers management service

task is only to collect and send data to the FSP, which has more computing resources than the node-level producer and uses interconnection technologies such as Wi-Fi, Cellular, LoRa, etc. The semantic data annotation service is responsible for data collection, aggregation, modelling, mapping linking and querying. These functions will be introduced later.

1) *Data organization in FSP*: The main objective of the data organization mechanism is to collect and formulate raw data sent by IoT producers by filtering out redundant data and converting the rest to XML, JSON, CSV format, or relational databases (RDB), etc, and aggregating of filtered data to valid high-quality information. Therefore, minimizing the computing resources required for the annotation processes. The RDB, CSV, XML, JSON parsers, then, receive these aggregated high-quality data to process further by modeling these aggregated data to make the relationship of data objects. This modelled data is converted into a Resource Description Framework (RDF) and linked with semantic annotations using the ontology. The resource RDF can be described by statements that allow the definition of attributes and values. The RDF uses a triplet to describe different elements in a statement. A triplet (subject-predicate-object) is described as a node-arc node whose direction flows from subject to object. For example: *"The FSP is a microservice architecture"* or *"The FSP contains many services"*.

- *Subject* is an element to identify the subject in a statement or is the resource which is the source of an arc in an RDF model.
- *Predicate* is an element that defines the attribute of the subject in the statement or the property part of a triple.
- *Object* is an element that determines the value of an attribute, it can be a resource (URI), or a value (Literal).

A set of linked triplet creates a graph-based RDF model, nodes in the graph can be *Subject* or *Object*, and arcs is *Predicate*. Thus, converting from a variety of data formats to the RDF format need to preserve the integrity and meaning of data. In this paper, we propose an algorithm allowing the conversion of relational database model to new data model using in Semantic Web technology, which is represented by RDF graph with triples. In the DBB, each triplet corresponds to a row in a table, and the values refer to each column.

Algorithm 1: Converting RDB to RDF graph

Input : RDB T
$PK : primarykey, FK : foreignkey$
Output: RDF graph G
Procedure `convertRDB2RDF()`
  $S \leftarrow T.size()$
  for $i \leftarrow 1$ *to* S
    $C[i] \leftarrow$ `ClassRDF(`T`)`
    $S_r \leftarrow T.Row[i].size()$
    for $j \leftarrow 1$ *to* S_r
      $R \leftarrow T.Row[j]$
      `Triple()` $\leftarrow Tp$
      $Tp.Subject \leftarrow R.id$
      $S_c \leftarrow R.Cells.size()$
      for $k \leftarrow 1$ *to* S_c
        if *(*`ColHead(`*R.Cells[i])* *not in* (PK, FK)*)*
          $P \leftarrow$ `Predicate(ColHead(`*R.Cells[i])*`)`
          $Tp.Predicate[k] = P$
          $O \leftarrow$ `Object(Value(`*R.Cells[i])*`)`
          $Tp.Object[k] = O$
        if *(*`ColHead(`*R.Cells[i])* *in* FK*)*
          foreach tp *in* $RelatedTable(T)$
            $P \leftarrow T.Name \cup tb.Name$
            $Tp.Predicate[k] = P$
            `Triple(`*tb*`)` $\leftarrow$ `ColHead(`*R.Cells[i]*`)`
            $URI = tb.Subject$
            $Tp.Object[k] = URI$
      $G.AddTriple(Tp)$
  return G

When converted to the RDF diagram, the classes (Class) correspond to each table, each row in the table is represented by a statement in the form of triples (S, P, O) to form the RDF diagram, where a subject (S) is the key attribute value, predicates (P) correspond to the labels of each column, object (O) is the value at the column corresponding to the predicate (P). Algorithm 1 represents a detailed transformation from a relational database to an RDF graph.

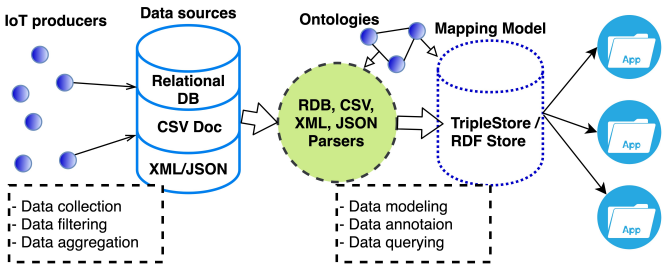

Fig. 4: The semantic data annotation service

2) *Data annotation based on ontology*: handling such large-scale heterogeneous data and processing it in real-time will be a key factor towards building smart applications. Ontology-based semantic approaches have been used to solve these issues related to large-scale heterogeneity and interoperability. The ontology describes technical details that are necessary for a producer registration, producer connection, as well as producer data provisioning, and can also be used as meta-data source by other microservice. The semantic annotation service processes each producer received data from the parsers as illustrated in Fig. 4 afterwards, it implements the annotation process to tag these data by defined concepts from domain ontology. The output of this step is a TripleStore database.

To enable efficient storage and retrieval of this information, we reuse ontologies based on the FIESTA IoT ontology [3], which is a unified existing ontology for IoT. It provides most of the concepts identified in [4] with criteria on core concepts when designing an ontology. We borrow the concepts of *Sensor*, *Service*, *Location* and build an ontology for the producer and consumer domain and extend them with new core concepts such as *Producer*, *Consumer*, *Interface*, *Protocol* and *Network* to describe objects in heterogeneous environments and the interoperability of different IoT multi-network access domains.

3) *Producers access service*: To manage a heterogeneous set of producers and their different native protocols. In the first step of the service, the producers need to be registered are defined by their profiles. Each producer profile contains a unique identifier of the producer to be registered. Detailed information of producers is not necessary, because they are contained in the producer ontology. We define and describe a profile of producer shown in Table II.

TABLE II: Producer profile

Attribute	Required/Obtional	Remark
`producerId`	required	Producer identifer
`manufacturer`	required	Producer manufacturer
`name`	optional	Producer name
`location`	optional	Producer Coorditates
`model`	optional	Producer model
`description`	optional	Producer description

The producer register procedure is performed and illustrated in Fig.5 by following steps: (1) the producer joins to the FSP node by asking information about the domain of FSP via a DNS service; (2,3) The DNS service responses based on the received information with an IP address of the FSP node; (4) the producer, then send their profile to register with the FSP node via this IP address; (5) The FSP node forwards this profile to the semantic data annotation service to process and update the producer's profile into a corresponding format; (6) the FSP node confirms the successful registration and wait for the producer send data to the FSP node.

4) *Consumer access service*:

Similar to the producer access service mentioned above, in this section, we provide the management mechanism of consumers through a service called consumer access service.

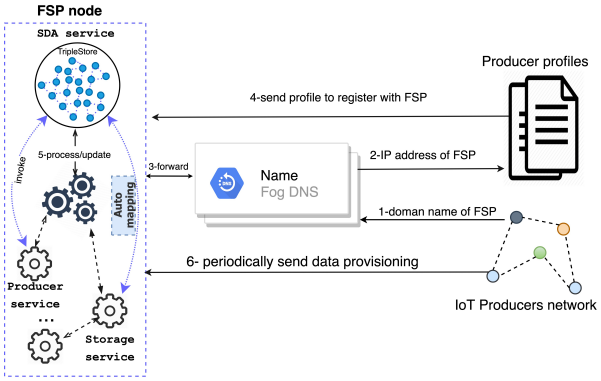

Fig. 5: IoT producers registration service

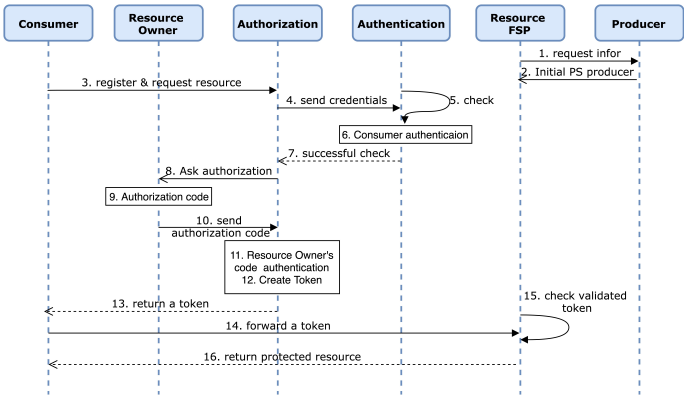

Fig. 6: Registration and access procedures for consumers

Consumer entity as defined in section III. It is a software, an API, a service or an application that allow users to interact with it to obtain information from producers. We use the concept *Consumer* which defined in the ontology with attributes and the relationships with data objects. This concept enables to access resources in the FSP node through authenticating and authorizing mechanism. To register into the FSP architecture, we also use a profile of *Consumer* as basic information to become a member of the FSP, illustrated as in Table. III.

TABLE III: Consumer profile

Attribute	Required/Obtional	Remark
`name`	required	Consumer name
`company`	required	Company name
`email`	required	Email address
`postalcode`	required	Postal code
`mobilenumber`	required	Mobile number

Fig. 6. illustrates a sequence diagram related to the registration procedure and request resources with FSP. This procedure is authenticated and granted through two Authentication and Authorization mechanisms after it has been successfully registered with consumer's profile information.

The procedure starts when the consumer wants access to the available resources in the IoT producer network. First, it sends a request to the *Authorization service* to asks for authentication. In turn, *Authentication service* checks the validity of consumer by asking the consumers to provide a credential (`username` and `password`), then the consumer contacts *Resource Owner* of the resource. The *Resource Owner* grants access to the client by sending an authorization code. This authorization code delivers to the *Authorization service* to verify and release a token which contains the details of the consent provided to the *Consumer*. The *Consumer* forwards the token to the *Resource FSP* to check the validity of the received token and provide the protected resource. However, this part is out of the scope of the paper due to the pages limitation.

IV. Experimental Testbed

In order to demonstrate an experimental testbed, as shown in Fig. 7, we have been developed to practically implement the main functionalities of the proposed architecture such as register procedures of producer and consumers and manage its information in the core concepts using the ontology. The IoT producer network includes Raspberry Pis 3 Model B with an operating system called *Hypriot* and built-in WiFi, GPIO ports, MicroSD card slot, memory, video/audio outputs. Hypriot [10] enables the execution of Docker on a Raspberry Pi and deploys services in Docker runtime. Those are also reasons why we use Docker technology to deploy distributed services. The RasPis will run a Python script to collect data from a DHT11 sensor (i.e., temperature, humidity) and send it to the FSP via CoAP protocol.

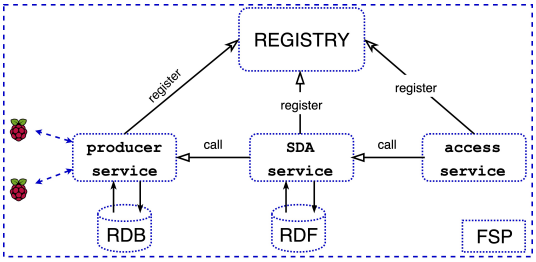

Fig. 7: Evaluation Setup

We also show the example result of mapping from RDB to RDF graph through *Algorithm 1* with related tables in a producer management relational database. The primary key of relational tables is the bolded properties, as described in the following:

PRODUCER(**producerID**, producerName, producerModel, producerManufacturer)

`{90035229,iotproducer1,Raspberry B 3,UK }`
`{90035230,iotproducer2,Raspberry B 3,FR }`

SENSOR(**sensorID**, sensorName, sensorType)

`{sen01, DHT11, Temperature }`
`{sen02, DHT22, Humidity }`

OBSERVATION(**observationID**, **producerID**, timestamp)

`{obsen01, 90035229, 30/4/2020 }`
`{obsen02, 90035230, 29/4/2020 }`

DETAILOBSERVATION(**observationID**, **sensorID**, location, value)

```
{obsen01, sen01, RoomLiving, 20 C }
{obsen01, sen02, Kitchen, 70 % }
```

The data converted from an RDF graph into the data model for semantical annotation is stored as a Triple-Store, as shown in Table. IV. The namespace `fsp = http://www.fsp.com/fspdb/` indicates that the elements with the `fsp` prefix belong to a triplet (S, P, O). The execution of the query data in the RDF store is supported by SPARQL language.

TABLE IV: The RDF store triplet

Subject	Predicate	Object
fsp/producer/90035229	fsp/producer/producerID	90035229
fsp/producer/90035230	fsp/producer/producerID	90035230
fsp/sensor/sen01	fsp/sensor/sensorID	sen01
fsp/sensor/sen01	fsp/sensor/sensorName	DHT11
fsp/producer/90035229	fsp/producer/producerName	iotproducer1
...	...	...

Table. V and Table. VI are API endpoints between the FSP node and microservices. The former depicts the endpoint to be used for processing the register producer's profile in the FSP. The later is the endpoint to be applied to access the resource by an external user.

TABLE V: Producer register service

Request/Response	
URL	POST http://fsp:8080/fspdb/register
JSON	{"id":6,"serial":"90035229","name":"iotproducer1","model": "Raspberry B 3","manufacture":"UK"}

TABLE VI: Resource access service

Request/Response	
URL	GET http://fsp:8080/fspdb/resource/access
JSON	[[27, "RoomLiving", "20", 4, 6],[33, "Kitchen", "70", 5, 7]]

V. Conclusion

This paper introduced a unified data model for a microservice architecture for IoT heterogeneous devices based on fog computing paradigm and given several particular notations such as components, services, fog nodes, producers and consumers and services of FSP. We have presented the design and proposal the fog computing service architecture for managing heterogeneous producers and consumers technologies through applying semantic web technologies with helping of ontology and registration procedures and access control. Besides, we develop a data converting algorithm for a unified data model. In the future, we will extend full-fledged fog computing architecture with the capability handling diversity and accessible multi-network to support interoperability in various IoT domains.

References

[1] OWL - Semantic Web Standards. https://www.w3.org/OWL/.

[2] Adhatarao, S. S., Arumaithurai, M., and Fu, X. FOGG: A Fog Computing Based Gateway to Integrate Sensor Networks to Internet. In *2017 29th International Teletraffic Congress (ITC 29)*, vol. 2, pp. 42–47.

[3] Agarwal, R., Fernandez, D. G., Elsaleh, T., Gyrard, A., Lanza, J., Sanchez, L., Georgantas, N., and Issarny, V. Unified IoT ontology to enable interoperability and federation of testbeds. In *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*, pp. 70–75. December 2016.

[4] Bajaj, G., Agarwal, R., Singh, P., Georgantas, N., and Issarny, V. 4W1H in IoT Semantics. 65488–65506, Conference Name: IEEE Access, 2018.

[5] Bonomi, F., Milito, R., Natarajan, P., and Zhu, J. Fog Computing: A Platform for Internet of Things and Analytics — SpringerLink. In *Bessis N., Dobre C. (Eds) Big Data and Internet of Things: A Roadmap for Smart Environments. Computational Intelligence.*, vol. 546, Springer, Cham, pp. 169–186. 2014.

[6] Bonomi, F., Milito, R., Zhu, J., and Addepalli, S. Fog Computing and Its Role in the Internet of Things. In *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, MCC '12, ACM, pp. 13–16. 2012.

[7] Chien, Y. H., and Lin, F. J. Distributed Semantic Reasoning Enabled by Fog Computing. In *2019 International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pp. 1033–1040. July 2019.

[8] Faruque, M. A. A., and Vatanparvar, K. Energy Management-as-a-Service Over Fog Computing Platform. 161–169. April 2016.

[9] Gia, T. N., Rahmani, A. M., Westerlund, T., Liljeberg, P., and Tenhunen, H. Fog Computing Approach for Mobility Support in Internet-of-Things Systems. 36064–36082. IEEE Access, 2018.

[10] hypriot. Docker Pirates ARMed with explosive stuff · Docker Pirates ARMed with explosive stuff. https://blog.hypriot.com/.

[11] Iorga, M., Feldman, L., Barton, R., Martin, M. J., Goren, N. S., and Mahmoudi, C. Fog Computing Conceptual Model. 03.2018.

[12] Jabbar, S., Ullah, F., Khalid, S., Khan, M., and Han, K. Semantic Interoperability in Heterogeneous IoT Infrastructure for Healthcare. 2017.

[13] Kharel, J., Reda, H., and Shin, S. An architecture for smart health monitoring system based on fog computing. *Journal of Communications 12* (04 2017), 228–233.

[14] Okafor, K. C., Achumba, I. E., Chukwudebe, G. A., and Ononiwu, G. C. Leveraging Fog Computing for Scalable IoT Datacenter Using Spine-Leaf Network Topology. *Journal of Electrical and Computer Engineering 2017* (2017), 1–11.

[15] Rahman, H., and Hussain, M. I. Fog-based semantic model for supporting interoperability in IoT. 1651–1661. 2019.

[16] Saurez, E., Hong, K., Lillethun, D., Ramachandran, U., and Ottenwälder, B. Incremental Deployment and Migration of Geo-distributed Situation Awareness Applications in the Fog. In *Proceedings of the 10th ACM International Conference on Distributed and Event-Based Systems*, DEBS '16, ACM, pp. 258–269.

[17] Singh, S., and Chana, I. QoS-Aware Autonomic Resource Management in Cloud Computing: A Systematic Review. 42:1–42:46. ACM Comput. Surv, 2015.

[18] Skarlat, O., Schulte, S., Borkowski, M., and Leitner, P. Resource Provisioning for IoT Services in the Fog. In *2016 IEEE 9th International Conference on Service-Oriented Computing and Applications (SOCA)*, pp. 32–39.

[19] Soldatos, J., Kefalakis, N., Hauswirth, M., Serrano, M., Calbimonte, J.-P., Riahi, M., Aberer, K., Jayaraman, P. P., Zaslavsky, A., Žarko, I. P., Skorin-Kapov, L., and Herzog, R. OpenIoT: Open Source Internet-of-Things in the Cloud. In *Interoperability and Open-Source Solutions for the Internet of Things*, I. Podnar Žarko, K. Pripužić, and M. Serrano, Eds., Lecture Notes in Computer Science, Springer International Publishing, pp. 13–25. 2015.

[20] Stojmenovic, I. Fog computing: A cloud to the ground support for smart things and machine-to-machine networks. In *2014 Australasian Telecommunication Networks and Applications Conference (ATNAC)*, pp. 117–122.

[21] Toor, A., ul Islam, S., Ahmed, G., Jabbar, S., Khalid, S., and Sharif, A. M. Energy efficient edge-of-things. 82. 2019.

[22] Verma, P., and Sood, S. K. Fog Assisted-IoT Enabled Patient Health Monitoring in Smart Homes. 1789–1796. June 2018.

[23] Yi, S., Hao, Z., Qin, Z., and Li, Q. Fog Computing: Platform and Applications. In *2015 Third IEEE Workshop on Hot Topics in Web Systems and Technologies (HotWeb)*, pp. 73–78.

[24] YI, S., QIN, Z., AND LI, Q. Security and Privacy Issues of Fog Computing: A Survey. In *Wireless Algorithms, Systems, and Applications*, vol. 9204, pp. 685–695. Springer International Publishing, Cham, 2015.

[25] ZHAN, Z.-H., LIU, X.-F., GONG, Y.-J., ZHANG, J., CHUNG, H. S.-H., AND LI, Y. Cloud Computing Resource Scheduling and a Survey of Its Evolutionary Approaches. In *ACM Comput. Surv.*, vol. 47, pp. 63:1–63:33. July 2015.