A unified and semantic data model for fog computing

Hoan Le, Khaled Boussetta, Nadjib Achir

A unified and semantic data model for fog computing

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

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

Nadjib Achir
L2TI, Université Sorbonne Paris Nord,
Villetaneuse, France
Inria, Palaisau, France
nadjib.achir@univ-paris13.fr
nadjib.achir@inria.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 in terms of devices, communication technologies, protocols, data formats, and semantics. Data generated from diverse sources use different semantics and models. It makes semantic interoperability as one of the outstanding issues in provisioning seamless communication and services over various IoT platforms. Semantic models are a useful approach for exchanging semantically annotated information between such heterogeneous applications. Seamless communication between different types of applications is usually achieved by using middleware, ontology, semantic web technologies, and there are no unified storage and management methods for information on IoT devices. In this paper, we propose a solution for handling IoT data's heterogeneity and facilitating interoperability and contextual information management. The solution comprises a semantic data model for the generic description of elements in our proposed fog computing platform, namely Fog Services Provider (FSP), to support IoT. Besides, this data model is designed for managing any device using different communication technologies that are fully described and formalized in an ontology format called *FSPontex*.**

***Index Terms*—Internet of Things, heterogeneity management, semantic, context information, data model, ontology.**

I. Introduction

Internet of Things (IoT) is a collection of connected physical objects accessible through the internet. It enables the connection of things that can be monitored and remotely controlled via a wireless infrastructure. They can be assigned IP addresses and can transmit data over a network without human intervention to gather data from the environment that better perceives nature. The importance of IoT is increasing in all fields over the years. The number of connected devices on the Internet will exceed 50 billion by 2020, this according to Cisco [6]. By 2022, 1 trillion networked sensors will be embedded in the world and up to 45 trillion in 20 years. Therefore, the automated device monitoring reduces human interaction and improves productivity.

IoT devices are widely diverse primarily because of the different hardware and operating systems used. The heterogeneity of the information provided by the underlying devices is one of the most highlight features of the IoT domain. These heterogeneous devices communicate through a variety of protocols at low power networking (ZigBee, ZWare, LoRaWAN, and Bluetooth), or traditional networking protocols (Ethernet, Cellular, WiFi), and even application protocols such as CoAP (constrained application protocol), MQTT (message queuing telemetry transport), XMPP (extensible messaging and presence protocol), and AMQP (advanced message queuing protocol). These protocols are designed for domain-specific applications with particular features. Moreover, these devices are expected to be deployed in different areas of applications to observe the environment and generate enormous data continually—differences in the data formats, types, or syntax results in interoperability issues between the applications. Due to the lack of common understanding between different devices and platforms with resource constraints in IoT, it is not easy to understand the exact meaning (semantics) of the exchanged content. Thus, much work has to be done to ensure interoperability.

The traditional IoT service model's paradigm 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 these service providers' approaches, the IoT domain can be viewed as vertical silos of different IoT applications without horizontal connectivity between them. This lack of interoperability with independent services, poses a threat to the overall adoption and acceptance of the IoT domain, especially for applications that can profit from multiple devices. One possible solution is to achieve semantic interoperability in a heterogeneous IoT environment is to use semantic annotation of raw sensing data using; for instance, the ontology approach. In general, all features of the semantic model are deployed centrally in the cloud. While the computing capabilities of the cloud are essential, 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 data types. Concurrently, providing a unified data model to support interoperability in IoT by using Semantic Web (SW) technologies.

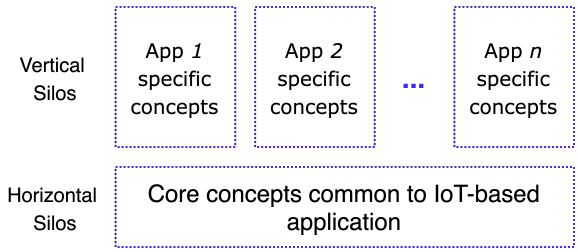

Fig. 1: Core concepts for IoT domains [8]

In the literature, fog computing-based architectures and applications are available and can be considered as potential solutions to the heterogeneity problems [11], [14]. However, semantic-based approaches are not commonly applied in the fog computing architectures.

To achieve semantic interoperability, we proposed a simple fog-based unified and semantic data model in our previous work [15]. In this extended version, we have improved that by developing a semantic-based annotation approach to handle such large-scale heterogeneous data and process for latency-sensitive applications by using a common unified ontology. It should describe the core concepts common to all IoT applications (i.e., horizontal) and concepts that are specific to applications (i.e., vertical), as illustrated in Fig 1, and is defined by using the 4WH1 methodology [8]. The rest of the paper is organized as follows: Section II presents related works. Section III introduces our proposed fog services provider (FSP) architecture for IoT and manages IoT devices through defining concepts, properties, and relationships between data objects with the help of ontology. Section IV provides more detailed data information to model elements with ability semantic data annotations in the FSP platform. 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 data models for IoT. The W3C Semantic Sensor Networks Incubator Group (SSN-XG) [4] addresses the issues of heterogeneity in sensor networks with concepts mentioned to describe sensors, observations, and related concepts. However, it does not describe domain concepts, time, location and real-time data collection issues, etc. To overcome the spatial and temporal problem, *Xue et al.* [20] proposed an ontology to describe heterogeneous sensors with concepts sensor types included *Advanced_Sensor* and *Normal_Sensor*, along with sensor capabilities *static*, or *dynamic*. They also introduce concepts to solve issues of the sensor management and data sharing in a sensor network. However, their ontology supports only a small number of sensors and provides semantic support for a limited number of sensor features. Furthermore, their concept of location is limited to building rooms and floors.

IoT-O ontology [18] defines modular extensible ontologies for modeling some of the main concepts in IoT. They focus on two sets of requirements *Conceptual* and *Functional* to form the core of any IoT-related ontology. Where the concepts are based on the description of devices, data, services, and life cycle. Moreover, *Functional* requirements are defined to enable the semantic community, which can follow best practices. *IoT-O* is one of the first approaches towards the unification of IoT ontologies by reusing concepts from SSN [4], (SAN, DUL, QUDT) [1], oneM2M and define some new concepts. However, it lacks a complete view of context and leaves out another important concept related to the monitoring and management of IoT devices. *Bermudez-Edo et al.* [9] introduced *IoT-Lite* by extending concepts from SSN ontology [4]. This ontology also added interoperability between sensor data among heterogeneous platforms while keeping the semantics lightweight. Besides, IoT-Lite mainly focuses on the minimum concept discovery and relationships that can respond to a maximum number of application queries. It also follows the linked data approach for data and knowledge. However, it has not included actuator and RFID, which are an integral part of today's IoT application.

M3 ontology proposed by *Gyrard et al.* [13] is an effort to integrate different IoT-related concepts from different ontologies into a unified taxonomy that describes concepts such as *Domain of Interest*, *Physical Phenomena*, and *Units of Measurement*. However, M3 is heavy; a lightweight version of M3 is M3-Lite [2] and has proposed to complement device modeling and service concepts. But, it lacks several aspects related to context and policy. The Light-weight Ontology (LiO-IoT) takes up a challenge in the range of IoT ontologies by introducing sensors, actuators, and RFID as IoT concepts [17]. It borrowed some concepts like *Sensors*, *Objects*, and *Coverage* from existing ontologies (SSN and IoT-Lite) to help in the provision of semantic interoperability. The authors have performed experiments to evaluate the LiO-IoT ontology to verify the round trip time of a query and compare it with both IoT-Lite and SSN. The results have shown that it has a similar response time with IoT-Lite and a better performance than SSN. However, it is not mentioned to the context information and architectural components related to communication technologies.

Our work aims at designing a unified data model with the capability of context information sharing and facilities that manage IoT heterogeneous devices using different technologies. We also investigate interoperability between devices instead of data and the Semantic Web to facilitate sensor discovery.

III. Our proposed Fog Services Provider (FSP)

As mentioned in our previous work [15], some basic notions related to entities such as *Producer*, *Consumer*, and *FSP (Fog Services Provider) node* are core elements to form our proposed fog computing architecture as shown in Fig. 2. Where, *Producer* and *Consumer* elements are objects that directly interact with the FSP node via some protocols (MQTT, HTTP, or WebSockets, etc,.). The *Producer* can be physical (sensors, actuators, smartphones) or non-physical entities (virtual device, web services) that uses to generate data and need

to be addressable in the network. The *Consumer* has a play role as a software entity that enables the analysis and process of data generated from *Producer*. The FSP node element is the heart of fog architecture. It enables rapid development, management, and scaling of IoT projects. The FSP platform is built based on the advantages of a micro-services model (*self-discovery*, *loose coupling*, and *high cohesion*) with a components-based architecture and are divided into several their functionalities.

The first component is the *IoT producers*, including a set of heterogeneous things with different multi-network access technologies (LoRA, ZigBee, Bluetooth, Cellular, etc.). It is data sources that emitted data with various formats. Next is a component of *Security/Authentication & Authorization management*. It provides mutual authentication and encryption at all points of connection so that data is never exchanged between producers and FSP Platform without a proven identity. The *Access* component enables IoT producers to connect, authenticate, and exchange messages with this FSP platform using the MQTT, HTTP, CoAP, or WebSockets protocols. The *Registry & Config* component establishes and manages metadata such as the producers' attributes and capabilities. It assigns a unique identity to each producer that is consistently formatted regardless of the type of producer or how it connects, and it is also responsible for handling and maintaining information hardware configuration, computation requirements, and connectivity of producers requested by various applications. The *Monitoring* component always keeps track of system performance and resources, services, and responses. It helps choose the appropriate resources during operation. The *Resources management* component maintains the allocation of resources, scheduling, and deal with energy-saving issues. The *Analytics* component includes several functionalities such as data analysis, data flow, rule engine. At the data analysis stage, acquired data are analyzed, filtered, data trimming and reconstruction are also done when necessary. Following the processing of the data, the data flow component decides whether the data needs to be stored locally or sent to the cloud for long-term storage. The Gateway is responsible for supporting of different type of protocols such as MQTT, CoAP, HTTP, etc, and providing access to heterogeneous Producers and Consumers. The functionalities of the FSP include:

- Provision and handle heterogeneous producers,
- Collect, analyze, and visualize data form producers,
- Support interoperability for different kind of producers.

Besides, the FSP can model processed data to make semantic contexts and provide interoperability at data annotation level through concepts, properties, and the definition of relationships between data objects by the *RDFs & OWL* and *Storage* components. 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 [3]). To manage a different set of producers and their different native protocols, we have experimented

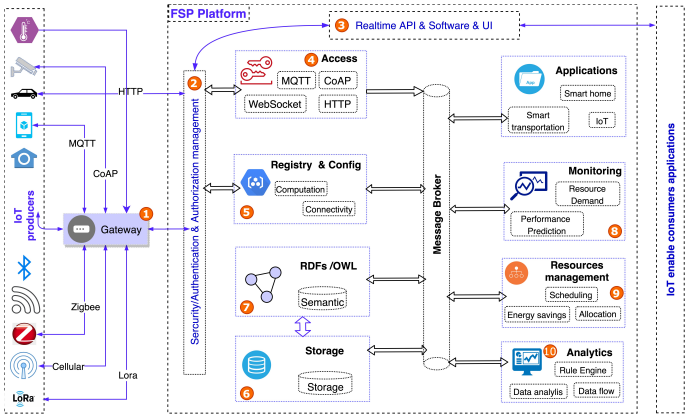

Fig. 2: Microservices architecture of Fog computing

with two procedures for registering a new producer and mapping raw data that generated from producer into semantic data. To register, the *Producer* sends its profile information (`producerId`, `manufacturer`, `name`, `location`, `model`, `description`) to the FSP. This information is used to map into a triplestore to store for purpose further. However, this proposed fog computing architecture still lacks some elements to create a unified data model with capacity interoperability and context information sharing for various IoT application domains. In this paper, we define a complete data model to overcome the mentioned issues and describe more details in the next section.

IV. Proposed Data Model for FSP

This section provides more detailed data information to model elements in the FSP platform. These elements are core concepts to build a *semantic data model* that enables the sharing of context information and interoperability criteria.

A. Core concepts

Establishing a complete unified ontology for IoT could be a challenge, as there are more than 200 domain ontologies [12]. There are specific concepts for most ontologies inherent to IoT application domains, while all IoT platforms share some of the concepts used. To provide a contextual data information model, we reuse the ontology based on the FIESTA-IoT ontology [7], which is an existing unified ontology for IoT. It provides most of the concepts identified in [8] with criteria on core concepts when designing an ontology. However, it lacks concepts for annotating context information to share knowledge and has a limited notion for *Software*, *Hardware*, and *Communication*.

We borrow the concepts of *Sensor*, *Service*, *Location, Observation* and build an ontology called FSPontex and extend them with new core concepts, including *Producer*, *Consumer*, *InterfaceNetwork*, *Communication*, *Network*, and *ContextEntity* to describe objects in heterogeneous environments and the interoperability of different IoT multi-network access domains. The *ContextEntity* concept is the root entity to describe the contextual information of a single node. It can be a context

for smart cities, smart health, or smart home, etc. It is a superclass of *Platform*, which is a representative entity for entities that can be attached. This entity is a direct superclass of two context classes, namely *Consumer*, *Producer*.

1) *Platform*: is a core component of FSP architecture. It represents a fog node or an FSP node that provides services, components, and modules for *Producer* and *Consumer*. It is also responsible for managing data information and sharing context information with other entities. Moreover, this element enable to model processed data to make semantic contexts and provide interoperability at data annotation level through concepts, attributes, and the relationships between data objects.

2) *Producer*: is designed to perform a particular task such as event detection or changes in its environment. It consists of two elements: hardware, on behalf of a physical entity like *Sensor*, *Actuator*, *SmartPhone*, and software is a computational data element representing a physical entity, including *API*, *Virtual Device*, *Webservice*, or *Microservice*.

3) *Consumer*: is a software entity that enables analysis, process data generated from *Producers*. It can be a user's applications that provide some authentication and authorization mechanisms to allow access from outside through a *Service* concept class.

4) *InterfaceNetwork*: is the point of interconnection between a *Producer* and a private or public network. This concept enables to manage heterogeneous communications of *Producer* either both low-layer networking (ZigBee, Bluetooth, LoRaWAN, etc.) and high-layer networking (MQTT, CoAP, XMPP, or AMQP) protocols.

5) *Communication*: describes the state of a Producer's communication via protocol stack. The state can be in terms of the general quality, efficiency, security, frequency, or availability of communication to provide appropriate contexts and share information for other applications.

6) *Network*: this concept class provides the state of IoT platform's network based on exchanged *Communication* contexts as well as from deployed network management for IoT producers. Increased awareness of the state of the network can offer to more effective solutions. Especially those resulting from inherent constraints (e.g., resource constraints).

B. Properties, relationships, and annotations

Handling such large-scale heterogeneous data and processing it in real-time will be a key factor in building smart applications. Ontology-based semantic approaches have been used to solve these issues related to large-scale heterogeneity and interoperability. Semantically annotating IoT data is a fundamental step toward developing smarter and interoperable IoT applications. The principal information model focuses on modeling data generated from producers and the relationships between elements in the FSP. Each element is described by classes, properties, and relationships. These classes reflect the core concepts as mentioned in the previous section.

As depicted in Fig. 3, the central class that the other classes directly link to is the *Producer*. This abstraction represents a data stream originating from an IoT data producer. It has

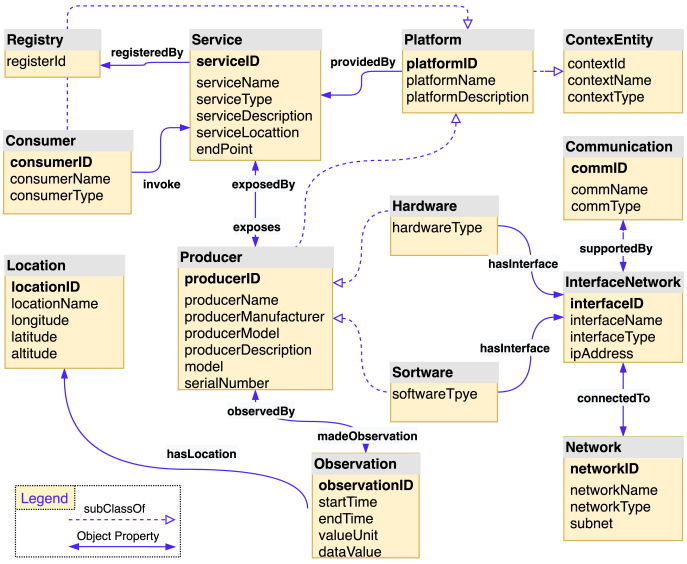

Fig. 3: Data model for Fog Services Provider

datatype properties, object properties, and annotation properties that capture events or changes information of the IoT data that would mainly be used for reference rather than actual consumption by an application.

- Datatype property: is used to assign data values for an entity. It can be of type *boolean*, *date*, *int*, etc. If we denote *A* as an attribute and **xsdIRI**(*A*) is the IRI (Internationalized Resource Identifier) of the data type *xsd* corresponding to the data type of attribute *A*. For example, a *Producer* entity has attributes and datatypes such as:

```
producerId xsd:int,
producerName xsd:string,
producerManufacturer xsd:string,
producerModel xsd:string,
producerDescription xsd:string,
serialNumber xsd:byte,
isMobile xsd:boolean
```

- Object property: is made to represent relationships between objects. Unlike databases and object-oriented programming languages, properties in OWL are defined independently of classes. When they are used, objects are identified as belonging to the class (domain) and value (range) of the properties. For example, a described relationship between *Producer* and *Observation* is a complementary combination by defined object properties that are *observedBy* property and *madeObservation* property, respectively.
- Annotation property: allows to add annotations on individuals, class names, property names, and ontology names. It is responsible for explaining the relationship between the classes and establishing the relations between the data required for efficient machine processing. It may make the information more readable for data analysts and the object of an annotation property must be either a data literal, a URI reference, or an individual. Fig. 4 illustrates an instance of a data model with a combination of object properties, datatype properties, and an-

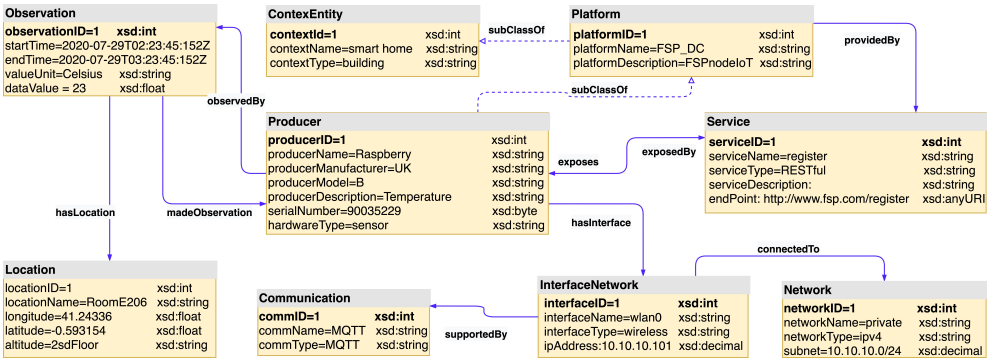

Fig. 4: An example of a data model annotated with the proposed ontology

notation properties. The example below shows the *Annotation Property* for two objects *Producer* and *Observation*:

```
<owl:AnnotationProperty
    rdf:about="&fsp;observedBy">
<rdfs:label xml:lang="en">observed
    by</rdfs:label>
</owl:AnnotationProperty>

<owl:AnnotationProperty
    rdf:about="&fsp;madeObservation">
<rdfs:label xml:lang="en">made
    observation</rdfs:label>
</owl:AnnotationProperty>
```

C. Dealing with IoT Interoperability issues

To provide interconnectivity and interoperability, we divide interoperability issues into three different interoperability levels are network layer interoperability, communication interoperability, and interoperability at data annotation level. These levels are performed in six steps such as data collection, data filtering, data aggregation, data modeling, data annotation, and data querying. Where, the first three steps are responsible for analyzing and formulating raw data send by *Producers* by filtering out redundant data, removing duplicated and unnecessary data. These three steps are represented for network layer interoperability level and communication interoperability level with attending of entities such as *Producer*, *InterfaceNetwork*, *Communication*. At the data annotation level implements the annotation process to tag data that processed previously by concepts through three steps later. These steps model data by the domain ontology and reference ontologies. Semantic annotation of *Producer* data by utilizing a standard mechanism and vocabulary can provide interoperability between IoT vertical silos. Therefore, it can be exploited by other services. At this level *Service*, *Consumer*, *Observation*, and *Platform* entities have an important role to form FSPontext with the interoperability and share context information along with their relationships.

D. Description language and query in Data Model

The most used and well-known language to describe ontologies is Ontology Web Language (OWL) proposed as a standard by W3C's Web Ontology Working Group [3]. Part of this work is conducted using the Protégé tool [Protégé, version 5.5.0]. It is developed by the Stanford Research Center based on Java language. One important advantage of Protégé is the higher compatibility with different ontology description languages such as WebOnto [10] and OntoEdit [19]. Beside, Protégé enables users to build, edit classes and properties, import different ontologies, visualize ontologies in various techniques, reasoning ability, create rules, and execute queries using a configurable graphical user interface (GUI). Fig. 5 illustrates the relationship between classes in the FSPontex.

By providing semantic annotation to producer data, the FSPontex ontology is stored in a single *RDF* or *OWL* file that is light-weight and supported by SPARQL query language [5]. This language allows users to write queries against what can loosely be called "key-value"data or, more specifically, data that follow the RDF specification of the W3C. Thus, the entire database is a set of `subject-predicate-object` triples. This is analogous to some NoSQL databases' usage of the term "document-key-value", such as MongoDB. The following query returns names and a serial number of every *Producer* in the dataset:

```
PREFIX fsp:
    <http://www.fsp.com/fsp/ontologies/fspontex#>
SELECT ?producerName
       ?serialnumber
WHERE
  {
    ?producer p                 fsp:Producer .
    ?producer fsp:producerName  ?producerName .
    ?producer fsp:serialnumber  ?serialnumber .
  }
```

This query joins together all of the triples with a matching `subject`, where the type `predicate`, "p", is a *producer* `(fsp:Producer)`, and the *Producer* has one or more names `(fsp:producerName)` and serial number `(fsp:serialnumber)`.

E. Context information in the Modeling step

In our ontology, the *ContextEntity* concept is used to address different contexts, such as a generic context and a specific

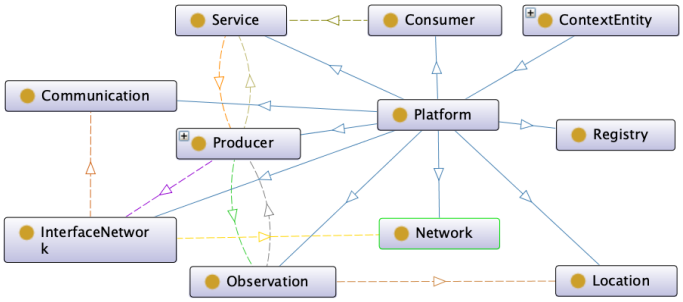

Fig. 5: A visualization functionality of the FSPontex

context. The generic context can be deployed in any smart context-aware system, while the specific context is used for a specific smart domain, as a smart city, smart health, etc. The *contextType* property provides information to identify for these contexts. Therefore, to provide context information, a system must follow some steps as defined by [16], including Acquisition, Modeling, Reasoning, and Distribution. The Acquisition refers to collect the raw data from an IoT producer, from the environment. The Modeling process transforms the data in a specific format to turn its input for the Reasoning step. One of the techniques for Modeling is using ontology. The Reasoning is the main step in the context. It uses different data enrichment methods (e.g., a set of rules, probabilistic, data fusion). The Distribution is responsible for spreading the context information. In this paper, we focus on the Modeling step, and it is important to convert the context into a predefined format. An instance to share information of *Producer* in a smart home application with its profile information, as shown in Table. I.

TABLE I: Sharing specific producer's profile into a generic context

Specific Producer Profile	**Generic context**
`name: raspberry`	
`manufacturer: UK`	
`model: B`	HighTempLevel
`description: room temperature`	InsideE206Room
`serialnumber:90035229`	AffernoontPeriod
`temperature of room: 36°C`	
`date time: 2020-07-29T14:23:45:152Z`	
`longitude: 41.24336`	
`longitude: -0.593154`	
`altitude: 2sdFloor`	

V. Conclusion

Interoperability in IoT is difficult to achieve due to its heterogeneous nature and the lack of standard architecture. The available IoT ontologies are not adequate for semantic interoperability when an interaction between devices is limited to a specific domain of IoT. However, ontologies designed in various contexts are not able to validate the semantic interoperability between heterogeneous IoT devices. This article presents a view on how to achieve interoperability at the data and knowledge levels to support smart applications in IoT domains. We have proposed a new semantic data model for fog computing platform with help of the ontology, named *FSPontex*, and defined several new concepts to build a complete ontology, including *Producer*, *Consumer*, *Communication*, *InterfaceNetwork*, *Network*, and *ContextEntity* which enables sharing contextual information between IoT applications. The FSPontex is an easily accessible and understandable semantic model and can apply for IoT supported platforms. In the future, we will need to be conducted to finalize the validation of our ontology and to investigate its more complex test scenarios.

References

[1] Linked Open Vocabularies. https://lov.linkeddata.es/dataset/lov/vocabs.

[2] M3-Lite, https://lov.linkeddata.es/dataset/lov/vocabs/m3lite - Oct 2018.

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

[4] Semantic Sensor Network Ontology. https://www.w3.org/TR/vocab-ssn/.

[5] SPARQL Query Language for RDF. https://www.w3.org/TR/rdf-sparql-query/.

[6] Afshar, V., Evangelist, C. D., and Salesforce. Cisco: Enterprises Are Leading The Internet of Things Innovation. Aug, 2017.

[7] 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.

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

[9] Bermudez-Edo, M., Elsaleh, T., Barnaghi, P., and Taylor, K. IoT-Lite: A Lightweight Semantic Model for the Internet of Things. In *2016 Intl IEEE Conferences on Ubiquitous Intelligence Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld)*, pp. 90–97.

[10] Domingue, J., and Keynes, M. TADZEBAO AND WEBONTO: DISCUSSING, BROWSING, AND EDITING ONTOLOGIES ON THE WEB. 21.

[11] Giang, N. K., Blackstock, M., Lea, R., and Leung, V. C. Developing IoT applications in the Fog: A Distributed Dataflow approach. In *2015 5th International Conference on the Internet of Things (IOT)*, pp. 155–162.

[12] Gyrard, A., Bonnet, C., Boudaoud, K., and Serrano, M. LOV4IoT: A Second Life for Ontology-Based Domain Knowledge to Build Semantic Web of Things Applications. In *2016 IEEE 4th International Conference on Future Internet of Things and Cloud (FiCloud)*, pp. 254–261.

[13] Gyrard, A., Datta, S. K., Bonnet, C., and Boudaoud, K. Cross-Domain Internet of Things Application Development: M3 Framework and Evaluation. In *2015 3rd International Conference on Future Internet of Things and Cloud*, pp. 9–16.

[14] Le, H., Achir, N., and Boussetta, K. Fog computing architecture with heterogeneous Internet of Things technologies. In *2019 10th International Conference on Networks of the Future (NoF)*, pp. 130–133.

[15] Le, H., Achir, N., and Boussetta, K. Fog services provider architecture for IoT. In *2020 11th International Conference on Networks of the Future (NoF)*.

[16] Perera, C., Zaslavsky, A., Christen, P., and Georgakopoulos, D. Context Aware Computing for The Internet of Things: A Survey. 414–454.

[17] Rahman, H., and Hussain, M. I. LiO-IoT: A Light-weight Ontology to provide Semantic Interoperability in Internet of Things. 571–575.

[18] Seydoux, N., Drira, K., Hernandez, N., and Monteil, T. IoT-O, a Core-Domain IoT Ontology to Represent Connected Devices Networks. In *20th International Conference on Knowledge Engineering and Knowledge Management - Volume 10024*, EKAW 2016, Springer-Verlag, pp. 561–576.

[19] Sure-Vetter, Y., Erdmann, M., Angele, J., Staab, S., Studer, R., and Wenke, D. OntoEdit: Collaborative Ontology Development for the Semantic Web. vol. 2342, pp. 221–235.

[20] XUE, L., LIU, Y., ZENG, P., YU, H., AND SHI, Z. An ontology based scheme for sensor description in context awareness system. In *2015 IEEE International Conference on Information and Automation*, pp. 817–820.