Managing Data Models in Broker-Based Internet/Web of Things Architectures

Pierfrancesco Bellini, Luciano Alessandro Ipsaro Palesi, Paolo Nesi
University of Florence, Distributed Systems and Internet Technology, DISIT Lab
https://www.disit.org , Https://www.snap4city.org, corresponding: paolo.nesi@unifi.it

In this context, an **IoT/WoT Platform** should abstract and manage all the entities/devices in the IoT Network, allowing to exploit them regardless of their position (connection with gateway/brokers), owner, protocol, format, etc. **Platforms** need to manage multiple IoT Networks and Brokers: some Brokers can be managed by third parties, e.g., the **External Brokers;** while the ones directly managed by the platform are called **Internal Brokers.** In realistic scenarios, third-party brokers are not setup and managed in terms of Device/Entity registration, subscription, data storage, search, etc., by the platform. As to **External Brokers**, entities/devices are registered on the broker, without providing notification to the connected platforms. On the contrary, a Platform should be able to recognize and manage device messages exchanged with any kinds of broker, any kinds of Device structure (which can be called the **Device Model**, for example the FIWARE Smart Data Models, SDM [1]) in order to register, process and store messages.
Recently, the Data Space concept has been introduced and it can be regarded as a generalization of the IoT Device concept [2], [3]. In the approach, a clear distinction from IDS (International Data Space) Metadata and messages is made, thus giving support for IDS Metadata Brokers.

The proposed solution is based on: (a) leverage interoperability reducing set up time to efficiently detect and learn how to process unknown data structures (devices, entities) distributed via brokers; (b) provision of data driven high rates in a broker-based platform, thus preserving full capability features of the data warehouse. To this end, an extension of the Snap4City Directory concept and tool has been created. The Directory is the main drive for interoperability in an efficient manner, and a number of other platform components serving the Directory are involved in obtaining the required performance to satisfy point (b). The solution supports: (i) Internal and External brokers, (ii) automated registration of devices/entities managed into External Brokers' single- or multi-tenant services, (iii) automated registration by harvesting and reasoning of data models/entities compliant with standard models such as FIWARE SDM, and any custom Data Model in Snap4City IoT Device Model providing a formal semantic definition of device attributes, (iv) fast data ingestion for ingesting / migrating historical data from legacy platforms and services to a new established uplevel platform, (v) sustained data usage from query demand and for data driven show changes in real time.

I. Architecture Overview for Interoperability

As reported in **Figure 1**, the **Directory** interacts with **Internal Brokers** to perform registration of devices/data flow (represented as D1, D2… Dn, sensors, actuators and data flow channels). It performs the semantic registration of device (data entities) into the **Knowledge Base**, KB (which is a semantic database RDF store) where all the entities and their relationships are modelled. The interoperable composition of data entities is guaranteed by the adoption of Km4City Ontology [4], [5] that creates a uniform layer abstracting from physical details and mechanisms needed to access them through different Brokers and usage of several data models and their validation, as well as semantic interoperability and matching. In the event of data lack, the KB provides knowledge to complete information on devices with the semantic part. In most Platforms, storage (including time series) is called Data Shadow and it allows to create some historical data of the Devices/Entities. In Snap4City, these data flow/messages can be produced by processes such as: **IoT App** (node-RED), Dashboards, and data analytics processes (in Python, Rstudio, etc.), etc.; *they are not described in **Figure 1***.

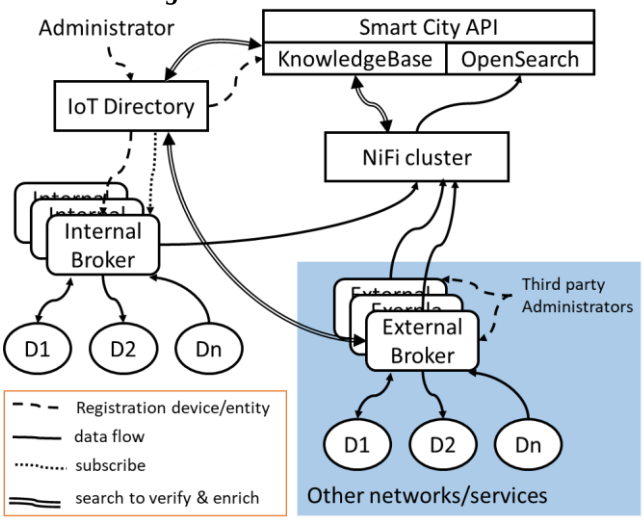

Figure 1 – Overview of the Architecture for data ingestion.

Snap4City models vs FIWARE models

FIWARE is a foundation which promotes open-source ORION Broker in standard NGSI. FIWARE includes a series of Generic Enablers software modules that perform functions in various IoT-based applications. FIWARE provides mechanisms for modelling and managing data and introduces RESTful NGSI API to interact with Orion Broker. According to NGSI V2 standard, each attribute has a name, a value and may provide its own metadata, and among the metadata, it may also define the unit of measure, the *unitCode*. In most SDMs, the *unitCode* is not defined leaving to data producers the choice to adopt some of its own. In the Snap4City model, each attribute has to be defined by the proprieties: Value Name, Value Type, Value Unit and Data Type. In this way, each device model attribute has a precise semantic formalization by name (e.g., V1), Value Type (e.g., Voltage), a specific unit of measurement by the Value Unit (e.g., V, mV, KV, vector of

mV values) and the Data Type clarifies the type of the data (e.g., *integer, float, string, json*). Please note that, Type in NGSI and Data Type of Snap4City refer to a different meaning. In particular, the NGSI Type is more generic than the Snap4City Data Type. For example, if the value is a number the NGSI Type can be "*numeric*", while a similar Snap4City Data Type can be "*integer*", "*float*" or "*double*". The FIWARE NGSI attribute definition is not specific enough to be processed by an inferential engine, due to its lack of semantic details; in fact, the NGSI *unitCode* is user defined and does not provide a unified semantic (does not belong to a common Dictionary), thus it may not be enough for the automated process. In the FIWARE usage of NGSI, the resolution is outsourced at application level. In Snap4City, the resolution is defined in a Dictionary (of KB) to conform any arrival message, to be faster and simpler in data ingestion and processing, while in FIWARE NGSI each message could change the *unitCode*.

Brokers' Registration

The first step to exploit the architecture as reported in **Figure 1** consists in the broker registration. As seen in the introduction, some platforms only provide support for a number of ready to use internal brokers. Snap4City allows the automated deployment of dedicated Orion brokers connecting them as Internal Brokers and it also supports the registration of External Brokers. In the event of External Orion Brokers, a given number of additional capabilities is possible. In the broker registration phase, several parameters are requested such as: endpoint, security, name, External/Internal, single/ multiple tenants, etc. Each broker is associated with a specific user or public, and each user belongs only to a single organization for security and privacy aspects [4], [6]. External Brokers are managed by third parties including their accessibility and usage. Other differences between Internal and External Brokers consist in the management of IoT Devices as explained hereafter.

Once a broker is registered, the IoT Directory automatically performs the data platform subscription (Ni-Fi) to the new broker for all its devices/topics, so that each new message generated by the broker would be directly brokered to data storage. On the other hand, this may not be true for the External Brokers since the IoT Directory/KB does not know all the entities/topics if they are not provided in the External Broker registration phase.

External Brokers and their Devices/Entities

To become easily interoperable with legacy brokers of third-party networks, we have defined a solution and process for the registration of External Brokers and their entities. At the first registration of an External Broker, thousands of devices should be discovered. In fact, Devices registered on a never connected **External Brokers**, are not registered on the IoT Directory and KB and as a consequence, Ni-Fi is not prepared to manage new data messages.

According to a ***faster approach***, we may suppose that the Snap4City Platform knows a set of Data Models (IoT device Models, FIWARE SDM, etc.). Subsequently, the harvesting process may recognize any device model (from a quick analysis of message format, device type and ID). In order to solve this problem, an ***automated harvesting approach*** of Devices/Entities on External Brokers has been. The registration of devices from External Brokers is one of the most innovative aspects addressed by IoT Directory which is capable of (i) harvesting brokers for device discovery, (ii) resolving semantic gaps on IoT device attributes/variables, (iii) registering devices, thus shortening the data ingestion and interoperability processes.

As to interoperability, the main identified and solved problems are those related to a large variety of Data Models coming from non-controllable External Brokers. The issue has been solved by designing and implementing a harvester and reasoner that is capable to automatically recognize/understand and map the new data models/types into those already known by Knowledge Base. This approach, together with the definition of a comprehensive meta model and dictionary, has allowed to speed up the process more than 800 times. Furthermore, the process is helped by Km4City ontology and Data Dictionary to recognize the new data types and models according to the semantic domain. Moreover, any processes of data discovery, registration and ingestion also impact on performance. To this end, the proposed solution has been assessed in terms of performance in harvesting brokers, discovering and registering devices, collecting messages and data access; thus, providing evidence of the maximum performance which can be obtained by each single front-end / back-end component/area and how they are influenced each other in the whole architecture.

Future work can be oriented on enforcing stronger encryption mechanisms which may impact on the protection of data and connections. An activity in this direction could be to investigate the enforcement of blockchain solutions on specific IoT devices.

REFERENCES

[1] F. Cirillo, G. Solmaz, E.L. Berz, M. Bauer, B. Cheng, E. Kovacs, A standard-based open source IoT platform: FIWARE, IEEE Internet of Things Magazine. 2 (3), (2019) 12–18.

[2] U. Ahle, J.J. Hierro, FIWARE for data spaces, Designing Data Spaces. (2022) 395. In: Otto, B., ten Hompel, M., Wrobel, S. (eds) Designing Data Spaces. Springer, Cham. https://doi.org/10.1007/978-3-030-93975-5_11.

[3] M. Jarke, C. Quix, Federated Data Integration in Data Spaces, Designing Data Spaces. (2022) 181. In: Otto, B., ten Hompel, M., Wrobel, S. (eds) Designing Data Spaces. Springer, Cham. https://doi.org/10.1007/978-3-030-93975-5_11.

[4] C. Badii, P. Bellini, A. Difino, P. Nesi, Smart city IoT platform respecting GDPR privacy and security aspects, IEEE Access. 8 (2020) 23601–23623.

[5] P. Bellini, D. Nesi, P. Nesi, M. Soderi, Federation of smart city services via APIs, in: 2020 IEEE International Conference on Smart Computing (SMARTCOMP), IEEE, 2020: pp. 356–361.

[6] C. Badii, P. Bellini, A. Difino, P. Nesi, G. Pantaleo, M. Paolucci, Microservices suite for smart city applications, Sensors. 19 (21), (2019) 4798.

[7]