Automated Sensor Registration, Binding and Sensor Data Provisioning

Pascal Hirmer[1], Matthias Wieland[1], Uwe Breitenbücher[2], and Bernhard Mitschang[1]

[1] Institute of Parallel and Distributed Systems
[2] Institute of Architecture of Application Systems
University of Stuttgart, Universitätsstr. 38, Stuttgart, Germany
{lastname}@informatik.uni-stuttgart.de

Abstract Today, the Internet of Things has evolved due to an increasing interconnection of technical devices. However, the automated binding and management of things and sensors is still a major issue. In this paper, we present a method and system architecture for sensor registration, binding, and sensor data provisioning. This approach enables automated sensor integration and data processing by accessing the sensors and provisioning the data. Furthermore, the registration of new sensors is done in an automated way to avoid a complex, tedious manual registration. We enable (i) semantic description of sensors and things as well as their attributes using ontologies, (ii) the registration of sensors of a physical thing, (iii) a provisioning of sensor data using different data access paradigms, and (iv) dynamic sensor binding based on application requirements. We provide the Resource Management Platform as a prototypical implementation of the architecture and corresponding runtime measurements.

Keywords: Internet of Things, Sensors, Ontologies, Data Provisioning

1 Introduction and Background

Nowadays, the integration of sensors becomes more and more important, especially for the emerging Internet of Things (IoT) [17] and Industry 4.0 [11]. Through the integration of raw sensor data, high level information can be derived that leads to huge benefits, e.g., in advanced manufacturing, smart homes or smart cities. In previous work [9], e.g., we presented SitRS – a service for situation recognition in smart environments based on raw sensor data. However, in this and in many other IoT approaches, sensors are manually registered and bound for processing, which is a complex and tedious task that requires technical knowledge about the sensors to be registered. Furthermore, adapters have to be manually created and deployed for each sensor to extract its data and to provision it to the application that is intended to consume the data. However, these steps are error-prone and can take hours or even days to be processed manually: a sensor expert has to configure the sensors, install a sensor gateway, bind the sensors, implement the

In: S. España, M. Ivanović, M. Savić (eds.): Proceedings of the CAiSE'16 Forum at the 28th International Conference on Advanced Information Systems Engineering, Ljubljana, Slovenia, 13-17.6.2016, published at http://ceur-ws.org

sensor data provisioning and establish interfaces to applications that intend to consume the sensor data. By doing so, he constantly has to communicate with domain-experts that want to build or use sensor-driven applications. In real-world scenarios, efficiency and accuracy are of vital importance. The drawbacks that come with a manual registration can lead to high costs due to occurring errors and a tedious, time-consuming registration process.

In this paper, our goal is to reduce the manual steps to the modeling of sensors and things using ontologies. All other steps (sensor binding, sensor data provisioning) can be processed automatically in milliseconds instead of hours or even days when conducting them manually. By doing so, we can reduce occurring errors that are more likely with manual processing and, as a consequence, save costs. To enable this, we need a means for automated sensor registration, sensor binding, as well as provisioning of the sensor data for further processing. In this paper, we present a method and system architecture for this means by (i) using ontologies for the definition of sensors and things, by (ii) enabling dynamic sensor binding through automated adapter deployment, and by (iii) sensor data provisioning using different data access paradigms. This enables direct *Machine-to-Application* communication by the abstraction of technical details.

Note that there are also objects in the world that are not observable by sensors. These objects are not covered in this paper.

The remainder of this paper is structured as follows: In Sect. 2, we describe related work. In Sect. 3, the main contribution of this paper is presented. After that, in Sect. 4, we evaluate the approach through runtime measurements of our prototypical implementation. Finally, in Sect. 5, we give a summary of the paper and describe future work.

2 Related Work

In [8] the goal is similar to our approach. The authors present a middleware called Global Sensor Network (GSN), which enables binding data sources like sensors and data streams with zero programming effort. To realize that, a virtual sensor abstraction is provided in [8], which allows declarative specification of deployment descriptors and basic processing of the data using SQL-like queries. In our approach, we separate these steps strictly. First, the dynamic sensor binding is done using ontologies. Second, the data is provisioned to sensor-driven applications.

Sensor description and configuration is standardized in IEEE1451.2 defining Transducer Electronic Data Sheets (TEDS) [12] that enable the self-description of sensors. Furthermore, an interface for standardized dynamic plug and play binding of sensors to networks is provided. In our approach, the physical binding of sensors is not the focus. We concentrate on an easy provisioning of sensor data to sensor-driven applications through the Internet. Note that standards such as the IEEE1451.2 could be used for sensor binding in our approach.

In [10], a REST-based interface is built to access sensors and retrieve their data. By doing so, this paper assumes an already in place sensor network bound to a gateway, which provides information of the sensors and manages their access for data retrieval. In contrast, our paper does not necessarily assume such a gateway and manages the sensor binding itself. Only the provisioning of sensor data is similar to our approach in [10].

In recent years, a large amount of Machine-to-Machine (M2M) gateways have been created such as FIWARE, OpenMTC[3], OpenIoT [16], or GSN [2,1]. These gateways serve as layer between physical sensors and "virtual" sensor data. It is important to note that the approach in this paper does not try to compete with these approved platforms but rather uses them, i.e., provides a more abstracted layer on top in order to enable an easy way to bind *things* in contrast to specific sensors, and to automatically provision data of the contained sensors to sensor-driven applications using Internet technologies. More precisely, the mentioned platforms can be used as gateways by our approach to realize the sensor binding.

Middleware between the physical and application layer gain more and more importance [4]. The main purpose of such middleware systems is to hide and abstract the physical details in order to allow the programmer to focus on the development of a specific sensor-driven application. Furthermore, it is important to abstract from the concrete environment and sensory that will be used after deployment of the application, in order to avoid a cumbersome and time-consuming configuration in each new environment.

SStreaMWare [7] is a service-oriented middleware for heterogeneous sensor data. It uses a hybrid approach supporting both centralized data streams and distributed sensor networks. Furthermore, a generic schema for sensor data representation is proposed (measures, timestamps and properties) and declarative queries can be executed on the sensor data streams. SStreaMWare has an approach similar to ours in managing the sensors and binding them based on the devices observed.

OntoSensor [14] is a sensor knowledge repository for modeling and management of sensors. It combines SensorML, IEEE SUMO, ISO 19115, OWL and GML. The goal of OntoSensor is to achieve a usage for description of sensors in different application domains. Through the combination of many different sensor definition languages, the ontologies become heavy-weight and complex. In our approach, we aim for a particularly lightweight ontology. Because of that, we decided to use only a subset of SensorML and omit using the whole OntoSensor ontology.

DCON [15] is an ontology for representation of user activity context. In [15], the authors combine many different OSCAF ontologies[4] to create a Personal Information Model: DDO (for Devices), DPO (for Presence), DCON [15] for representation of user activity context, and further. This is a specialized area and the ontologies are very detailed. In our approach the goal is to support any kind of domain, so the concepts are more generic. Not everything is focused on the

[3] www.open-mtc.org/ [4] http://www.semanticdesktop.org/ontologies/

users, in our approach things are the main focus and persons in contrast should not be monitored for privacy reasons.

In summary, the presented related work is mainly focusing on specific aspects like the access of sensors using gateways, or the execution of queries on sensor data streams or in a sensor network. However, the goal of our approach is to provide an easy-to-use ontology for the Internet of Things that combines sensor registration, binding of the sensors, and sensor data provisioning. Whereat the binding of a concrete sensor is done indirectly based on the things that are monitored by the sensors. Furthermore, our approach allows a *separation of concerns*, since the sensor data processing is specified separately, e.g., in the situation recognition as described in [9], based on situation templates that can be mapped onto different execution systems. Additionally, our approach allows the integration of heterogeneous sensor types in a standard way as REST resources or through a *publish-subscribe* model so that they can be accessed by multiple clients and in parallel.

3 Sensor Registration, Binding and Sensor Data Provisioning

This section presents the main contribution of this paper by introducing a system architecture and a method for ontology-based sensor registration, binding, and sensor data provisioning. Figure 1 depicts the overall architecture of our approach. It consists of the following main components: (i) the sensor registry, storing meta-information about the physical things and sensors, (ii) the sensor ontology, containing sensor binding information, (iii) the sensor adapters, extracting the data from the sensors, that can be deployed directly on a thing or on an adapter platform, and (iv) the *Resource Management Platform (RMP)* provisioning the sensor data as remotely accessible resources (pull) or via a publish-subscribe approach (push).

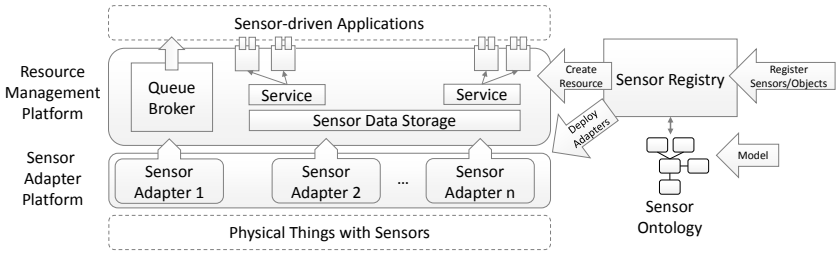

Figure 1. Architecture for ontology-based sensor registration and binding

This architecture is applied through a method that enables automated sensor registration, binding and sensor data provisioning in five consecutive steps. Note that a full automation of this method is possible.

Step 1: Task Definition

- **Input:** thing identifier, sensor identifier(s), *optional:* ontology snippet
- **Output:** thing identifier, sensor identifier(s), *optional:* ontology snippet

In the first step of the method, the things and sensors to be registered are defined by so called *task definitions*. Each task definition contains a unique identifier of the thing to be registered and, if specific sensors of a thing should be registered, also the sensors' identifiers. Detailed information of sensors and things are not necessary, because they are contained in the sensor ontology (cf. Step 2). In case a thing or a sensor is not known, i.e., is not represented in the ontology, an ontology snippet describing their properties has to be added to the task definition, which will be processed in Step 2. However, in the following, we assume that the ontology contains all sensors and things of the specific domain our approach is applied to and is modeled *correctly*.

Step 2: Ontology Traversal

- **Input:** thing identifier, sensor identifier(s), *optional:* ontology snippet
- **Output:** list of sensor specifications, thing identifier, sensor identifier(s)

Based on the information contained in the task definitions of Step 1, additional, specific information about things and sensors are retrieved from the ontology next. The ontology describes technical details that are necessary for an automated sensor registration, sensor binding, as well as sensor data provisioning, and can also be used as meta-data source by sensor-driven applications. These information include sensor specifications (accuracy, frequency, ...), information about the sensor access, i.e., about sensor binding in terms of the corresponding adapter in a sensor adapter repository, and information about the contained sensors of a thing. To enable an efficient storage and retrieval of these information, we use ontologies based on the XML-based sensor markup language SensorML [6]. Ontologies offer a means for an automated editing and extension, e.g., using generated SPARQL queries. Furthermore, in Internet of Things scenarios, the use of ontologies is commonly accepted [14,13,3] due to the heterogeneous, dynamic environments that have to be integrated. On sensor registration, we traverse the ontology and search for the corresponding entry of the sensor or thing. Once the relevant sensor information is found, it is used for automated sensor binding, which is described next.

Step 3: Sensor Adapter Deployment / Automated Sensor Binding

- **Input:** list of sensor specifications, thing identifier, sensor identifier(s)
- **Output:** list of successfully deployed adapters

After the information necessary to bind the sensors has been extracted from the ontology, the next step is the automated sensor binding and, furthermore, the provisioning of the sensor data. To enable this, we first need a means to extract

the sensor data from the corresponding sensors. This requires adapters, which are connecting to the sensors' serial interfaces, extract the data as a stream, and send it to the Resource Management Platform (e.g., using HTTP or MQTT). An advantage of our approach is that the adapters do not have to care about sensor data provisioning to sensor-driven applications, because they send the data directly to the centralized RMP that manages the provisioning for them.

Sensor adapters are deployed automatically. First, the adapters are retrieved from a repository and can be parameterized (e.g., with the RMP's URL). The information, which adapter is needed to bind the sensor(s) defined by the task definition is extracted from the ontology in Step 2. There are several possibilities how an adapter deployment can be realized: if the sensor is connected to a thing that is containing a powerful runtime environment such as, e.g., a Raspberry Pi, the adapter can be deployed directly using, e.g., SSH connections or more sophisticated approaches such as TOSCA [5]. However, in most cases this is not possible. Because of that, the adapters have to be deployed on external platforms, either self-implemented or using approved solutions, such as FIWARE or OpenMTC, that are capable to connect to the sensors, even if, e.g., they are embedded into a production machine, using Machine-to-Machine standards.

Step 4: Sensor Data Provisioning

- **Input:** list of successfully deployed adapters
- **Output:** REST resource URI(s), queue topic(s)

Once a sensor adapter is deployed and activated, it starts sending data to the RMP. However, the data can only be accessed by the sensor-driven applications after the fourth step is processed, the sensor data provisioning. In this step, the interfaces to the sensor-driven applications are established. The sensor data provisioning step represents the integration of all components, from the sensor adapters to the sensor data provisioning through the RMP. After the automated adapter provisioning (Step 3), the RMP is informed that the registered sensors have started sending their data. By doing so, entries in the sensor data storage as well as corresponding REST resources are created for each sensor to provision its data to enable the pull approach. Furthermore, we create topics in a queue for each sensor and publish these topics to the sensor-driven applications that can subscribe to them to enable the push approach. After this step, the sensor data are available to sensor-driven applications.

Step 5: Sensor Deactivation

- **Input:** thing identifier, sensor identifier(s)
- **Output:** list of successfully deactivated sensors

The last step is the deactivation of sensors once they are not needed anymore. To do so, the thing and the type of the sensor have to be provided to the sensor registry. Based on this information, the sensor registry finds running sensors of a thing with the corresponding type, connects to the adapters to terminate them,

clears the values from the sensor data storage, and removes the REST resources and the topics in the queue. Deactivation of sensors saves energy and, thus, costs.

4 Evaluation

We implemented an open-source prototype of the RMP that is in productive use within the project SitOPT. The sensor registry component is based on NodeJS[5] and offers a REST-based programmatic interface. The sensor registry uses SPARQL requests to access the sensor ontology, SSH to deploy the sensor adapters and HTTP to notify the RPM that a new sensor has been registered. Currently, the native file system is used as adapter repository. The ontology was defined using the Web Ontology Language (OWL) 1.1[6]. The access to the ontology is done by SPARQL requests through the Apache Jena[7] framework. The RMP is also implemented in NodeJS, which enables an easy definition of a RESTful interface. Furthermore, due to the lightweight platform, it offers high efficiency. The sensor data storage is implemented using the NoSQL database mongodb[8], which allows high efficiency, scalability, and data replication. The direct push approach was realized using MQTT[9] and the Mosquitto[10] broker.

We conducted runtime measurements of our prototype for evaluation purposes using a machine with a Core i5-3750K @3.4GHz and 8 GB RAM. We measured the average runtime of the steps described in the previous section based on 10 measurements: (i) the sensor registration took *1,91 ms*, (ii) the ontology traversal *6,73 ms*, and (iii) the adapter deployment *139,63 ms*. The measurements show that we could achieve the efficiency goals of this paper (cf. Sect. 1).

5 Summary and Future Work

In this paper, we present an approach for ontology-based sensor registration and sensor data provisioning. We introduce a system architecture and a method to make our approach applicable for a wide range of Internet of Things applications. After registration of a sensor, an adapter is deployed automatically that reads the sensor data, and passes it to the Resource Management Platform. The RMP creates topics in a MQTT queue and HTTP REST resources to provision the data. By doing so, we created an easy-to use solution for sensor-driven applications to bind sensors and access their data. Our goal was the registration, binding and sensor data provisioning in an automated manner to enable this within milliseconds in contrast to a manual processing of these steps that can take up to hours or even days. This goal was achieved as described in our evaluation.

In the future, we will add security, privacy and robustness features to our prototypical implementation and, furthermore, we will work on the performance that will supposedly decrease by adding these features. In addition, we will work on optimizations for our presented method.

[5] http://nodejs.org/ [6] http://www.w3.org/Submission/owl11-overview/
[7] https://jena.apache.org/ [8] http://www.mongodb.org/ [9] http://mqtt.org/
[10] http://mosquitto.org/

Acknowledgment. This work is partially funded by the DFG project SitOPT (610872) and by the BMWi project SmartOrchestra (01MD16001F).

References

1. Aberer, K., Hauswirth, M., Salehi, A.: A middleware for fast and flexible sensor network deployment. In: Proceedings of the International Conference on Very Large Data Bases (VLDB 2006) (2006)
2. Aberer, K., et al.: Zero-programming Sensor Network Deployment. In: Proceedings of the Service Platforms for Future Mobile Systems (2007)
3. Attard, J., Scerri, S., Rivera, I., Handschuh, S.: Ontology-based Situation Recognition for Context-aware Systems. In: Proceedings of the 9th International Conference on Semantic Systems (2013)
4. Atzori, L., Iera, A., Morabito, G.: The Internet of Things: A survey. Computer Networks (2010)
5. Binz, T., et al.: TOSCA: Portable Automated Deployment and Management of Cloud Applications, pp. 527–549. Springer (2014)
6. Botts, M., et al.: OGC Sensor Web Enablement: Overview and High Level Architecture. In: GeoSensor Networks. Springer Berlin Heidelberg (2008)
7. Gurgen, L., Roncancio, C., LabbÃĪ, C., Bottaro, A., Olive, V.: SStreaMWare: a service oriented middleware for heterogeneous sensor data management. In: International Conference on Pervasive Services (2008)
8. Hauswirth, M., Aberer, K.: Middleware support for the" Internet of Things". 5th GI/ITG KuVS Fachgespräch "Drahtlose Sensornetze" (2006)
9. Hirmer, P., Wieland, M., Schwarz, H., Mitschang, B., Breitenbücher, U., Leymann, F.: SitRS-A Situation Recognition Service based on Modeling and Executing Situation Templates. In: Proceedings of the 9th Symposium and Summer School on Service-Oriented Computing (SUMMERSOC 2015). pp. 247–258 (2015)
10. Ishaq, I., Hoebeke, J., Rossey, J., De Poorter, E., Moerman, I., Demeester, P.: Facilitating Sensor Deployment, Discovery and Resource Access Using Embedded Web Services. In: Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2012 Sixth International Conference on. pp. 717–724 (July 2012)
11. Jazdi, N.: Cyber physical systems in the context of Industry 4.0. In: Automation, Quality and Testing, Robotics, 2014 IEEE International Conference on (2014)
12. Lee, K.: IEEE 1451: A standard in support of smart transducer networking. In: Instrumentation and Measurement Technology Conference, 2000. IMTC 2000. Proceedings of the 17th IEEE (2000)
13. Probst, F., Gordon, A., Dornelas, I.: OGC discussion paper: ontology-based representation of the OGC observations and measurements model. Institute for Geoinformatics (ifgi) (2006)
14. Russomanno, D.J., Kothari, C.R., Thomas, O.A.: Building a Sensor Ontology: A Practical Approach Leveraging ISO and OGC Models. In: IC-AI (2005)
15. Scerri, S., Attard, J., Rivera, I., Valla, M.: DCON: Interoperable Context Representation for Pervasive Environments. In: AAAI Workshops (2012)
16. Soldatos, J., Kefalakis, N., Hauswirth, M., et al.: OpenIoT: Open Source Internet-of-Things in the Cloud. In: Interoperability and Open-Source Solutions for the Internet of Things. Springer International Publishing (2015)
17. Vermesan, O., Friess, P.: Internet of Things: Converging Technologies for Smart Environments and Integrated Ecosystems. River Publishers (2013)