Interoperability in the IoT - An Evaluation of the Semantic-Based Approach

Kristina Sahlmann
Institute of Computer Science
University of Potsdam
Potsdam, Germany
sahlmann@uni-potsdam.de

Florian Mikolajczak
Institute of Computer Science
University of Potsdam
Potsdam, Germany
florian.mikolajczak@uni-potsdam.de

Bettina Schnor
Institute of Computer Science
University of Potsdam
Potsdam, Germany
schnor@cs.uni-potsdam.de

***Abstract*—While the management of heterogeneous network devices is usually solved by protocols like SNMP and NETCONF, there is still no such accepted solution for the management of heterogeneous IoT devices. To avoid the vendor lock-in, several organizations like the IETF, W3C and ETSI are working on standards with a semantic-based approach. While the semantic approach seems to be appealing to solve the interoperability problem, there is still the question whether this approach is suited for constrained IoT devices. Herein, we present the evaluation of the MYNO, a semantic-based framework. MYNO is based on standards and open-source libraries and aims to support the management of constrained devices in the Internet of Things. We demonstrate the benefits of the semantic-based approach using a precision agriculture use case.**

***Index Terms*—Internet of Things, IoT, MQTT, Ontology, NETCONF, YANG, Interoperability, Semantic Web**

I. Introduction

Many vendor platforms for the Internet of Things (IoT) claim to be interoperable. But such platforms [1], [2] provide vertical solutions for IoT systems: All components like IoT devices, gateways, cloud-based services are from a single source. The disadvantage is the dependency on the vendor for enhancements or upgrades, the so-called *vendor lock-in*. Connecting two vendor solutions is near impossible.

Therefore, horizontal solutions are preferable. The idea is to decouple the IoT components from each other and use common standards for communication and data exchange. There are some initiatives for network management in the IoT but still there are many open issues (i.e. interoperability, scalability, security, energy saving) and no common comprehensive solution [3], [4].

Moreover, most vendor platforms are cloud-based. If the connection to the cloud is interrupted, local IoT devices might not operate properly. This is not acceptable for Industrial IoT solutions or solutions in the field of Smart Farming. The solution is *Edge Computing* where the managing edge node is deployed in the local enterprise network and accessible by devices even if the Internet connection is not available. Furthermore, the latency is decreased and the network bandwidth might be increased. If the edge node is connected to a cloud, for example for storing the data, network bandwidth can be saved, when the amount of data is reduced through pre-processing at the edge before transferring it to the cloud.

A network in the IoT contains often hundreds of constrained devices with sensors and actuators. Network configuration management tools are required to maintain such a network. One of the first IoT devices was a toaster in 1990, connected to the Internet by TCP/IP and managed by SNMP [5]. It had one control operation to turn the power on and off. The authors wanted to demonstrate the possibilities of SNMP. However, SNMP and the more recent NETCONF protocol [6] are not appropriated for running on constrained devices [7], [8].

Levels	Description	Examples
Level 4 Data Interoperability	Semantic data	OWL, DSL
	Syntactic data	XML, JSON
Level 3 Integration Interoperability	Application protocols	MQTT, COAP, HTTP
Level 2 Network and Transport Interoperability	Communication protocols	TCP/IPv6, UDP/IPv6, 6LoWPAN
Level 1 Physical Interoperability	Connectivity protocols	IEEE 802.15.4, WLAN

Fig. 1. Interoperability Model for the IoT

Interoperability can be achieved by using standards. However, *semantic* interoperability is necessary to achieve full interoperability. We propose an interoperability model, based on [9] and adapted for the IoT, as shown in Figure 1. There can be identified four levels which need interoperability: (1) the physical level which refers to the connectivity protocols like IEEE 802.15.4 or WLAN; (2) the network and transport level addresses communication protocols like TPC/IPv6, UDP/IPv6, 6LoWPAN; (3) the integration interoperability includes application protocols like MQTT [10], COAP [11], HTTP. Obviously, communication and application protocols are not sufficient to achieve interoperability in an IoT system. The meaning and the context of the exchanged data are also necessary to be defined. This needs annotations like ”what kind of device?”: a sensor, ”which functionality?”: provides measuring functionality, ”which kind of data?”: for temperature data, ”in which units?”: in degree Celcius or Fahrenheit. Such semantic annotations as wells as technical and non-technical attributes for data can be provided by machine-interpretable descriptions [12]. This is addressed by level (4) data interoperability which is subdivided into Syntactic Data and Semantic Data. Syntactic Data describes the format and

structure like XML and JSON; and Semantic Data describes the meaning of data and the common understanding of vocabulary, e.g. with the help of dictionaries, taxonomies, ontologies and formalization method for sharing meaning with a model language, e.g. OWL [13] or DSL [14].

To tackle the interoperability problem, the MYNO framework was proposed which is based on standards and open-source implementations and acts on the edge of the network [15]. The MQTT protocol is the common IoT protocol and the basis for the framework. The idea of the solution is to bridge the network configuration protocol NETCONF [6] and the MQTT protocol and enhance the architecture with semantics. The proposed MYNO framework is based on the following technologies: MQTT, YANG, NETCONF and Ontology, which build the acronym. YANG [16] is the data modeling language for the NETCONF protocol. An ontology is used for semantic device-descriptions. The MYNO framework pursues a semantic-based approach, supports interoperability for heterogeneous IoT devices, and provides a model-driven web client.

While the semantic approach seems to be appealing to solve the interoperability problem, there is still the question whether this approach is suited for constrained devices and thin edge nodes like a Raspberry Pi. Therefore, the contribution of this work is the evaluation of the semantic-based approach for the IoT to answer the following research questions:

- how are heterogeneous devices supported by the MYNO framework? (see Section III and IV)
- how to implement semantic device descriptions on constrained devices? (see Section IV)
- which ontology should be used for device descriptions? (see Section III)
- how much resources are needed on the edge node? (see Section V)
- does the system scale with the number of IoT devices? (see Section V)
- is this effort justifiable? Advantages vs. disadvantages should be considered. (see Section VI)

The paper is structured as follows: First, we outline the work at standardization organizations for interoperability in the IoT. Then, we present the semantic-based approach of the MYNO framework in Section III. The use case Precision Agriculture demonstrates in Section IV how new devices and sensors are integrated in the framework. The performance evaluation is presented in Section V. Finally, we discuss the results in Section VI.

II. Related Work

Organizations like the IETF, W3C, ETSI and Open Mobile Alliance (OMA) work on the standardization of an interoperability approach for the IoT [11], [17]–[19]. Most of them use a semantic-based approach. However, they often do not present a holistic framework. For example, the IETF focuses on protocols like COAP [11], the CoAP Management Interface (CORECONF) [20], and OMA on LwM2M [19]. ETSI is a member of the oneM2M [21] initiative and developed the SAREF ontology [18] for the IoT. oneM2M provides a technical specification which claims to achieve interoperability solutions for M2M and IoT technologies. However, it is focusing on the middleware services and does not consider the underlying network and devices.

Recently, W3C finalized two recommendations for the Web of Things (WoT): the WoT Architecture [17] and the WoT Thing Description (TD) [22]. The WoT is intended to enable interoperability across IoT platforms and application domains. Similar to the device descriptions in MYNO, the Thing Description is a vocabulary which describes an IoT device and provides common concepts for sensor observations, actions and events. They claim to follow the syntax of JSON-LD, which is supposed to enable extensions and rich semantic processing. However, the TD is a collection of many vocabularies and not a valid ontology according to a Semantic Web Standard like OWL. Therefore, semantic processing is not possible yet. Further, the vocabulary of the TD does not provide descriptions of the communication protocols. Instead, Binding Templates [23] were introduced to support different protocols like MQTT, CoAP, HTTP. However, from our experience, the vocabulary is not complete yet because some detailed constructs between protocols and device functionalities are missing.

The WoT Architecture outlines many use case patterns. A proprietary implementation is on the way[1]. While WoT defines security and privacy aspects by design, the architecture stays very high-level and is missing important aspects for implementation, e.g. a discovery process. The MYNO framework introduces a boostrapping process for discovery and has a proof-of-concept implementation (see Section III).

The IETF also identified the advantages of semantics and started to work on a Semantic Definition Format (SDF) [24] for data and interaction models in the IoT. The vocabulary is similar to the W3C TD and even smaller. It is using the JSON syntax and is not related to Semantic Web Standards. Therefore, the SDF has the same weaknesses: it is incomplete yet and semantic processing is not available.

A recent IETF draft specifies a mapping between YANG, the data modeling language for managed devices used by the network management protocol NETCONF, and the SDF format [25]. This is another indication that the MYNO approach is filling the gap, since it brings both worlds, network management and semantic, together. The NETCONF-MQTT bridge in the MYNO framework does exactly this: it translates the semantic-based device descriptions to the YANG model for the NETCONF protocol.

Obviously, there is a demand for semantic device descriptions and some organizations try to specify an architecture using such descriptions. However, MYNO specified a holistic framework and provides an implementation.

[1] https://projects.eclipse.org/projects/iot.thingweb

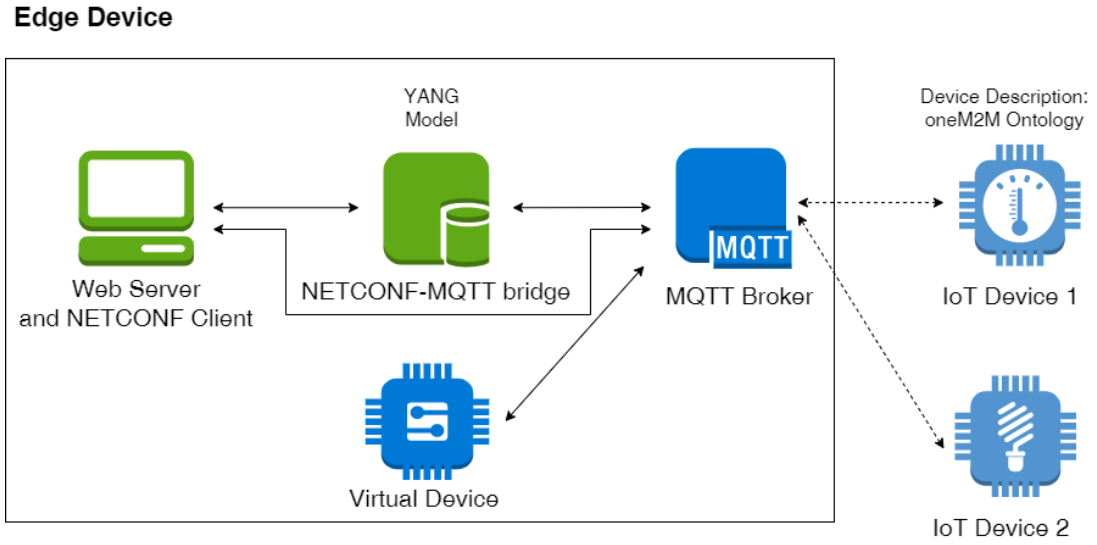

Fig. 2. MYNO System Architecture

III. MYNO: A Semantic-based Approach

A. Architecture

This section gives a short overview over MYNO's components and functionality. More details can be found in [26].

The MYNO framework consists of four components shown in Figure 2: a MQTT Broker, a NETCONF-MQTT bridge, a Web Client, and a Virtual device.

The MQTT Broker is the central part for the MQTT communication which is based on the publish-subscribe principle. Even constrained devices like the Texas Instruments CC2538dk [27] board (see Table I) with limited resources of computational power, memory, network bandwidth, and energy, support this simple protocol. They can subscribe to the so-called MQTT Topics and publish messages on such topics.

In the original network management architecture, a NETCONF server runs on the managed device and responds to RPC calls. Since such a NETCONF server has shown to be too heavyweight for IoT devices [8], we proposed the NETCONF-MQTT bridge which provides a connection between the NETCONF protocol and the IoT protocol, namely the MQTT protocol.

We introduced semantic device descriptions which provide the device capabilities [28]. The device descriptions are based on the oneM2M Base Ontology [29]. This ontology was originally chosen among many IoT ontologies for two reasons: (i) it is a small ontology for service and functionality description of devices which meets our requirements; and (ii) it is represented by the OWL 2, a Semantic Web Standard. Additionally, the SAREF ontology from ETSI is related to the oneM2M ontology and developed vertical domain ontologies. This might be a sign for a potential establishment of this ontology. We extended the vocabulary of the ontology to support the automatic generation of RPC calls by only four additional OWL classes (`YangDescription`, `AutomationFunctionality`, `ConfigurationFunctionality`, `EventFunctionality`) and two OWL Datatype Properties `mqttMethod` and `mqttTopic`. Using an ontology model ensures that common device capabilities are reusable, machine-interpretable and can be used for rich semantic processing.

A bootstrapping process defines how devices can join and leave a network managed by MYNO. This process has four CRUD operations and appropriate MQTT Topics. First, the devices publish their descriptions to the Topic `yang/config/create` during the create phase. The NETCONF-MQTT bridge processes them and generates a YANG model with RPC calls for actuators and descriptions for sensors. The YANG data model is used by the NETCONF protocol. On behalf of the NETCONF protocol, the IoT network at the edge, can be managed as a part of the entire enterprise network.

The Web Server acts as a NETCONF client and provides an graphical interface based on the generated YANG model (see Figure 3). This is a model-driven approach. A user can see which IoT devices are on the network and which capabilities they have. The grey fields represent the sensor values. The blue buttons are triggers for the actuators with parameters. The cyan buttons provide configurations of thresholds (if-then-conditions) which can trigger events, shown in yellow fields.

A virtual device is an optional component on the edge and enables the aggregation of IoT devices and sensor messages. This simplifies the implementation of applications like *"give me the mean temperature of all rooms on the south side"*. The virtual device is started on the edge node and subscribes to all bootstrapping topics and analyzes the device descriptions to collect controlling and measuring functions as well as configuration and automation functions. The virtual device publishes

its own device description to the MQTT broker and appears as a managed device in the bridge. This way, the virtual device integrates seamlessly into the MYNO architecture.

The MYNO source code is available as an open-source project[2].

B. Heterogeneous Devices

The feasibility of our approach with proposed scenarios and heterogeneous devices in terms of capabilities (sensors, actuators, etc.) and constraints was shown by several implementations. For new kinds of sensors or actuators, only the ontology-based device description was extended at minimum required. Since the YANG model is automatically generated from the device descriptions on the edge node, no additional apps from vendors are required. On the edge node, only the MYNO components are running: the bridge, the MQTT broker and optionally the virtual device. This approach is similar to the single source of truth (SSOT) architecture where data are only managed at one place. In MYNO, adding a new IoT device or new sensors requires only the update of one source, namely the update of the MYNO bridge source, but no source code from n different vendors has be installed on the edge node. This is different from smart home solutions where either we have a vendor lock-in or have to install apps from different vendors to manage a device.

TABLE I
CLASSIFICATION OF IOT DEVICES

Name	Example Devices (RAM, ROM/Flash)
Class 0	–
Class 1	CC2538dk (32 KB, 512 KB)
Class 2	ESP-32 NodeMCU (520 KB, 16 MB)
Beyond Class 2	Arduino Yún Rev 2 (16 MB, 64 MB), Raspberry Pi 3B (1 GB, 16 GB), Raspberry Pi Zero w (512 MB, 16 GB)

Currently, MYNO supports implementations for the Texas Instruments CC2538dk board [30], the Ardúino Yún [31] and the ESP-32 NodeMCU. Table I classifies the devices regarding RFC 7228 [32] based on their resources. The low-cost ESP-32 NodeMCUs were chosen for the scalability study (see Section V). The integration of these boards within the MYNO framework is discussed in the next section to demonstrate the usability of the MYNO approach.

IV. USE CASE PRECISION AGRICULTURE

For the evaluation of the MYNO framework, we setup a prototype implementation for IoT-based Precision Agriculture in a greenhouse. Precision Agriculture is made possible by the IoT and is an ongoing research field [33]–[36].

The requirements are: sensing environment data (air and soil), controlling irrigation, event configuration and notification when thresholds are reached, and finally, the automation of controlling functions (if-then condition).

A. Testbed

The prototype contains an edge component, a Raspberry Pi 3B, and 10 microcontroller boards which monitor 10 plants on the edge network. A single plant is representing a greenhouse or a field.

A WLAN hotspot is installed in the lab as an access point for the Raspberry Pi and the devices. As shown in Figure 2, the Raspberry Pi has running the MQTT broker from Mosquitto, the NETCONF-MQTT bridge and the web-based NETCONF Client as well as a Virtual Device.

The microcontroller boards are based on the low-priced ESP32 NodeMCU Module[3]. Every EPS32 board was extended through a breadboard equipped with sensors and actuators.

The following sensors are wired with the breadboard:

- A capacitive soil moisture sensor v1.2 determines the dielectric constant of the soil which is an indicator for dry or wet soil.
- Three sensors, namely temperature, humidity and air pressure, are combined in a GY-BME280 module which measures air condition.
- A raindrops sensor measures the conductivity of its surface. This is transformed into a binary output (rain/no rain) using an adjustable threshold.
- A GY-302 BH1750 light sensor measures intensity of visible light in lux.

The following actuators are deployed on the breadboard:

- A 5V mini water pump with external power supply (2 AA batteries) and watering pipe is controlled through the relais.
- A 1-relais 5V KY-019 module controls the water pump.
- A KY-016 RGB LED module is used for state signaling like a traffic light.

The power supply for a EPS32 board is ensured through a powerbank connected over the micro USB port.

B. Device Description

A device description must contain all capabilities which are provided by such an agriculture device. The following controlling functions for actuators are defined:

1) switch the RGD LED on and off;
2) switch the RGB LED with a given RGB color;
3) turn the water pump on and off;

For sensors, the device description was extended by reusing the OM-2 ontology [37] to provide units of measurements. The following sensor measurements are defined in the device description: soil moisture in percent, brightness in lux, air humidity in percent, air pressure in hectopascal, air temperature in degree Celsius, and raindrops detection in percent.

The full device description of the use case precision agriculture is given in the Appendix of [26].

For event configuration and notification, the device description was extended by new OWL classes. Such configuration defines a threshold value for a sensor as well as an interval

[2]https://github.com/ksahlmann/myno

[3]https://www.az-delivery.de/products/esp32-developmentboard

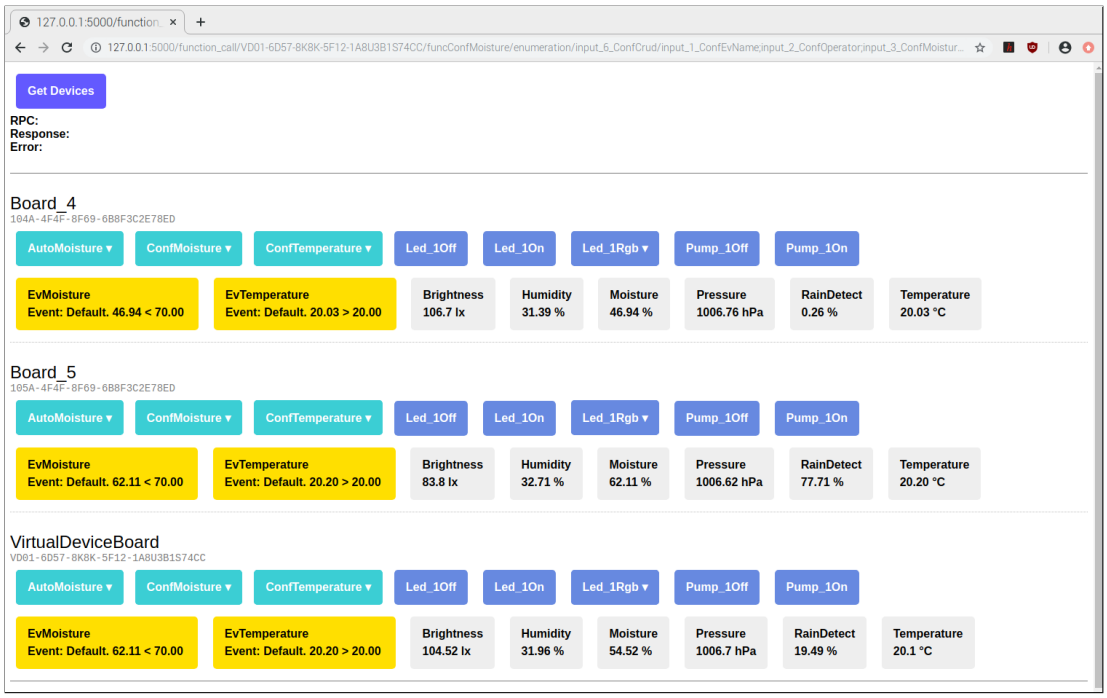

Fig. 3. Web-Client for MYNO

and duration for an event notification. Additionally, a name and a CRUD operation for this configuration must be defined. The difference to the controlling function is not only in the parameters (which are always the same) but also an MQTT Topic for publication of events like sensor values. The device description is reusing the TIME ontology [38] to provide ontology classes for interval and duration. The configuration functions are defined for two critical sensor measurements: soil moisture and air temperature. For example, events should be published every 10 seconds during the next 60 seconds when the soil moisture is under 30 percent.

The automation function (if-then condition) is a special case of event configuration and is defined in the device description as a combination of a configuration function and a controlling function instead of event notification. For example, if the soil is dry then turn the water pump on or switch the RGB LED to red to send a visual signal. Such automation functions can be used for event-based processing on a device instead of the event-based processing on the edge or in a cloud.

V. Evaluation

Potentially, the MQTT broker is scalable but the NETCONF-MQTT bridge behind must be fast enough to process device descriptions, sensor messages and RPC calls. We evaluated the scalability of the MYNO framework in our testbed with up to 10 devices.

In our experiment, we follow the guidelines from Jain [39]. The scalability evaluation considers only messages with device descriptions and sensor values. The actuator, configuration and automation messages are not considered because their occurrence is marginal.

The system parameters for the edge node, a Raspberry Pi 3B, and the ESP32 NodeMCU Module are shown in Table I. On the Raspberry Pi 3B, Raspbian OS 3.4 was installed, and the ESP32 NodeMCU devices were running with Arduino Sketch v. 1.8.10. All system components communicate through WLAN on 2.4 GHz basis. Each ESP32 device has 6 sensors which publish periodically their data. The sensors and their sensing interval are shown in Table II.

TABLE II
Sensing Intervals

Sensor	Interval [s]	MQTT Topic
Soil Moisture	600	sensor/moisture/moisture_1/UUID
Raindrops	300	sensor/rain/rain_1/UUID
Brightness	60	sensor/brightness/brightness_1/UUID
Air Temperature	60	sensor/temperature/temperature_1/UUID
Air Pressure	60	sensor/pressure/pressure_1/UUID
Air Humidity	60	sensor/humidity/humidity_1/UUID

The evaluation is divided into three sub-experiments: Device Description, Sensor Messages, and Energy Experiment; and answers the following questions:

1) Is the edge node capable to process the device description within acceptable time?
2) Does the MYNO framework scale with an increasing number of device descriptions and sensor messages?
3) Is the semantic approach feasible for constrained devices: What are the consequences in term of energy consumption?

A. Device Description Experiment

The goal of this experiment is to investigate the scaling of the MYNO framework during bootstrapping, when the devices publish their device descriptions. The processing of the semantic descriptions is implemented by the RDFLib v. 4.2.2 library [40] and this is the most computationally intensive part.

In this experiment, we vary the number of connected devices: 1, 3, 6, and 10 devices. The experiment for each number of devices was repeated three times.

The measured metrics are time for processing in the bridge, CPU and RAM usage in the Raspberry Pi (`vmstat`), and time for transmission (`tshark`).

We turned on the devices at 60 s intervals one after another. Since two devices are supplied from the same power bank, we started the devices pairwise in the case of 6 and 10 devices, again at 60 s intervals.

TABLE III
TIME FOR THE RDFLIB PROCESSING OF A DEVICE DESCRIPTION IN THE BRIDGE IN SECONDS (ROUNDED)

# Dev.	AVG	MAX	MIN	MEDIAN
1	12.556165	12.778631	12.437580	12.452284
3	8.551245	14.338671	5.430500	5.922269
6	8.116861	14.419649	5.372922	7.516252
10	7.564038	13.978273	5.624919	7.284434

Table III shows the time for processing a device description where the biggest part is the processing by the RDFLib library. The processing of the first device description takes much longer than processing the following descriptions. This behavior is always observable after a restart of the bridge. Obviously, there is some initializing work done by the RDFLib before the first RDF querying. But in the following, the median of the processing time increases only slightly from 5.9 up to 7.3 seconds.

The CPU load on the Raspberry Pi for starting 10 devices is shown in Figure 4. The CPU load increases drastically after receiving a device description but after the processing time it decreases again to the previous level. The green line shows the time spent running in kernel-mode, and the blue line shows the time spent running in user-mode. Thus, obviously RDFLib performs some kernel tasks. There are also some short peaks which show that the processing of device descriptions is a challenging task.

Figure 5 shows the RAM usage which remains under 500 KB and increases only slightly from 465 to 480 KB.

The transmission time of the device descriptions over WLAN is fast, but the values vary strongly. We observed a minimum transmission time of 26 ms (with 6 devices) and a maximum transmission time of 809 ms (with 3 devices). Each running device sends every minute 4 messages with sensor data, every five minutes 5 messages and every ten minutes 6 messages (see Table II). So, the influence of data collisions on the medium has a much higher influence on the transmission time of the device descriptions than the load on the edge node.

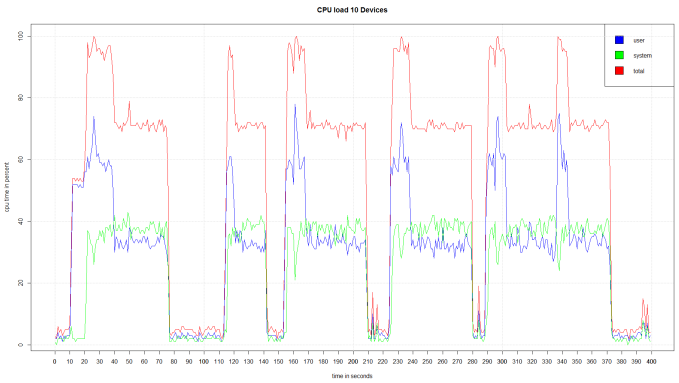

Fig. 4. CPU Load on the edge node: Starting 10 Devices.

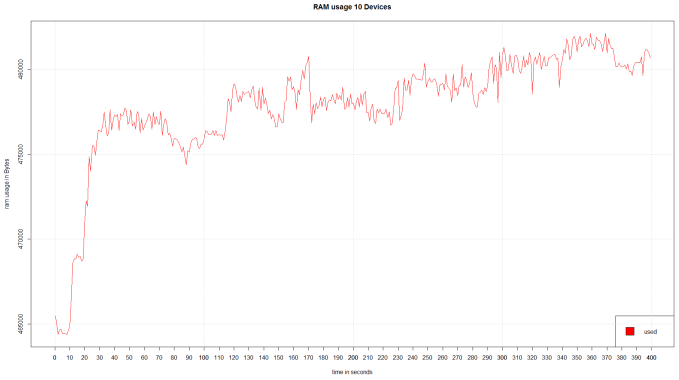

Fig. 5. RAM Usage on the edge node: Starting 10 Devices

B. Sensor Messages Experiment

The goal of this experiment is to check whether the system scales with the number of sensor messages. Therefore, we measure

- the transmission time of a sensor message, and
- the number of retransmissions.

The interval between the sensor messages is shown in Table II. The sensor messages are MQTT messages and consist of a topic and a sensor value. Hence, they are comparable in their size. The runtime duration is 1 hour per experiment. Each experiment was repeated three times.

Even when 10 devices are running, we observed fast transmission times between 74.4 and 127.5 μseconds. Reasons for these fluctuations are collisions on the medium and retransmissions on TCP level. Table IV shows the number of retransmissions monitored by the tshark tool running on the edge node. The number of these so-called *spurious*[4] retransmissions is low and increases almost linearly with the increased number of devices. Wireshark marks this kind of retransmissions as spurious since they are unnecessary: Wireshark has seen an ACK for the message, but the sender retransmits it. This is most probably a sign that the default timeout value within the

[4]https://www.chappell-university.com/post/spurious-retransmissions-a-concern

TCP communication stack on the ESP32 devices is too short in some rare cases.

TABLE IV
SENSOR MESSAGE RETRANSMISSIONS

# Dev.	Messages Arrived	Retrans.	[%]
1	89	2	2.2
3	261	5	1.9
6	521	11	2.1
10	842	26	3.0

C. Energy Experiment

Energy efficiency is important for energy-constrained devices i.e. powered by batteries. The energy consumption in this experiment is measured by the lifetime of the fully charged powerbanks (Schwaiger LPB220 533 powerbank with capacity of 2200 mAh).

Again, the sensors were configured as shown in Table II. We compare the lifetime of the devices for two different settings: with and without sleeping state between the messages. Without sleeping state, the powerbanks lasted for 16 h 36 min. This means that the boards shut down a part of the hardware, wake up, measure, publish values and sleep again 30 seconds. For this experiment, the lifetime of the powerbanks lasts much longer, namely 1d 17h 51m.

Further, we optimized some processes in the MYNO framework concerning the device descriptions to support energy efficiency by design. For example, the size of the device description of the agriculture use case is 37 KB. During bootstrapping and before sending the device description, a device sends a request to the NETCONF-MQTT bridge whether it is already registered on the network. This is a much shorter message and avoids an unnecessary transmission of the device description.

Additionally, the devices are configured with a compressed device description to reduce the energy costs for sending the message. In [41], the binary representations RDF HDT [42] and CBOR [43] were evaluated for the MYNO device descriptions. RDF HDT has shown much better space savings than CBOR in our use case. We observed space savings of 72.68% in RDF Turtle annotation and 84.06% in RDF N-Triples annotation because the input files have a verbose syntax. CBOR is not well suited for the compression of ontologies, since long strings, which are the main component of the device description, are not efficiently compressed by CBOR (less than 15 % savings in our example). But CBOR has of course its strength when sensor data have to be encoded for transmission.

VI. DISCUSSION OF RESULTS

This section discusses the pros and cons of the MYNO approach.

A. Interoperability

Regarding the interoperability model for the IoT shown in Figure 1, the results with the MYNO prototype are:

- Level 1: Physical Interoperability: demonstrated through implementations for IEEE 802.15.4 [30] and WLAN.
- Level 2: Network and Transport Interoperability: feasable through IP-based networks and protocols like IPv4/IPv6, 6LoWPAN [30] and TCP.
- Level 3: Integration Interoperability: achieved through application protocols MQTT and NETCONF.
- Level 4: Data Interoperability: achieved through OWL standard and ontology-based device descriptions, and the YANG model for the NETCONF protocol.

B. The burden of the semantic approach

While sensor data are typically only few Bytes in size, the size of a device description might be much bigger. For example, the size of the device description of the agriculture use case is 37 KB.

The performance evaluation on the precision agriculture use case shows that the edge node, namely the Raspberry Pi, is capable to process the semantic device-descriptions within acceptable time. Further, the MYNO framework scales with increasing number of devices and therefore its descriptions and sensor messages.

The experiments have shown that the semantic approach is feasible for constrained devices also in terms of energy consumption. The bootstrapping process prevents unnecessary transmissions of device descriptions. Thus, in best case it will be transmitted only once when a device joins a network.

The data transmission over MQTT in a WLAN network worked well as the performance evaluation has shown. However, it is challenging to transmit several kilobytes of data in a 6LoWPAN network. Hence, bigger messages must be sent in *slices* because of the constraint network bandwidth and memory on the device [30].

VII. CONCLUSION

We evaluated the semantic-based framework MYNO in the context of a high-precision agriculture use case. This work demonstrates that the semantic-based approach is suited for constrained IoT devices and also for an edge computing architecture.

Data interoperability was achieved through semantic device descriptions and the use of the de facto standard application protocol MQTT. The additional value of the ontology-based approach is the underlying model which represents the meaning of the data and which is self-descriptive and machine-readable. The YANG model for the management of new devices is automatically generated from the device descriptions. A further benefit of this approach is shown by the concept of the Virtual Device which is useful for the aggregation of device capabilities and sensor messages.

In the current version, MYNO uses RPCs for the implementation of the actuator operations to interact with the NETCONF client. In RFC 8040 [44], RESTCONF is specified which offers a REST-based interface to provide the actuator operations. Like the RPC variant, the REST API is defined

in the YANG model. Hence, MYNO framework can be easily extended with a REST interface.

Our experience with the MYNO framework shows that an IoT architecture for interoperability is an interdisciplinary project which requires knowledge in distinct fields like communication protocols, wireless sensor technologies, and semantics.

Acknowledgment

This research was funded by the Faculty of Mathematics and Natural Sciences of University of Potsdam in Germany. The responsibility for the content of this publication lies with the authors.

References

[1] AWS IoT. [Online]. Available: https://aws.amazon.com/de/iot/

[2] Azure IoT. [Online]. Available: https://azure.microsoft.com/en-us/overview/iot/

[3] J. d. C. Silva, J. J. P. C. Rodrigues, J. Al-Muhtadi, R. A. L. Rabêlo, and V. Furtado, "Management Platforms and Protocols for Internet of Things: A Survey," *Sensors*, vol. 19, no. 3, 2019.

[4] A. Čolaković and M. Hadžialić, "Internet of Things (IoT): A review of enabling technologies, challenges, and open research issues," *Computer Networks*, vol. 144, pp. 17–39, 2018.

[5] J. Romkey, "Toast of the IoT: The 1990 Interop Internet Toaster," *IEEE Consumer Electronics Magazine*, vol. 6, no. 1, pp. 116–119, 2017.

[6] R. Enns, M. Bjorklund, J. Schönwälder, and A. Bierman, "Network Configuration Protocol (NETCONF)," Internet Requests for Comments, Internet Engineering Task Force, RFC 6241, 6 2011.

[7] J. Schönwälder, K. Watsen, M. Ersue, and V. Perelman, "Network Configuration Protocol Light (NETCONF Light)," Working Draft, Internet Engineering Task Force, Internet-Draft, 1 2012.

[8] A. Sehgal, V. Perelman, S. Kuryla, and J. Schönwälder, "Management of Resource Constrained Devices in the Internet of Things," *IEEE Communications Magazine*, vol. 50, no. 12, pp. 144–149, 2012.

[9] W. Wang, A. Tolk, and W. Wang, "The Levels of Conceptual Interoperability Model: Applying Systems Engineering Principles to M&S," in *Proceedings of the 2009 Spring Simulation Multiconference*, vol. 168, San Diego, CA, USA, 2009, pp. 1–9.

[10] "MQTT Version 5.0," OASIS, Tech. Rep., 2019. [Online]. Available: http://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html

[11] Z. Shelby, K. Hartke, and C. Bormann, "The Constrained Application Protocol (CoAP)," Internet Requests for Comments, Internet Engineering Task Force, RFC 7252, 6 2014.

[12] P. Barnaghi, W. Wang, C. Henson, and K. Taylor, "Semantics for the Internet of Things: early progress and back to the future," *International Journal on Semantic Web and Information Systems*, vol. 8, no. 1, pp. 1–21, 2012.

[13] "OWL 2 Web Ontology Language Document Overview (Second Edition)," W3C OWL Working Group, W3C Recommendation, 12 2012. [Online]. Available: https://www.w3.org/TR/owl2-overview/

[14] Eclipse Vorto Language. [Online]. Available: https://github.com/eclipse/vorto/blob/master/docs/vortolang-1.0.md

[15] K. Sahlmann, T. Scheffler, and B. Schnor, "Ontology-driven Device Descriptions for IoT Network Management," in *2018 Global Internet of Things Summit (GIoTS)*, 6 2018, pp. 1–6.

[16] M. Bjorklund, "The yang 1.1 data modeling language," Internet Requests for Comments, Internet Engineering Task Force, RFC 7950, 8 2016.

[17] R. Matsukura, T. Kawaguchi, M. Lagally, K. Kajimoto, K. Toumura, and M. Kovatsch, "Web of Things (WoT) Architecture," W3C, W3C Reccommendation, 4 2020.

[18] L. Daniele, R. Garcia-Castro, M. Lefrançois, and M. Poveda-Villalon. (2019, 6) SAREF. [Online]. Available: https://saref.etsi.org/core/

[19] "OMA Lightweight M2M v1.1.1," Open Mobile Alliance, Specification 1.1.1, 6 2019.

[20] M. Veillette, P. van der Stok, A. Pelov, A. Bierman, and I. Petrov, "CoAP Management Interface (CORECONF)," Working Draft, Internet Engineering Task Force, Internet-Draft, 1 2021.

[21] "Functional Architecture," oneM2M, Technical Specification TS-0001-V3.19.0, 12 2019.

[22] T. Kamiya and S. Käbisch, "Web of Things (WoT) Thing Description," W3C, W3C Reccommendation, 4 2020.

[23] E. Korkan and M. Koster, "Web of things (wot) binding templates," W3C, W3C Working Group Note, 1 2020.

[24] M. Koster and C. Bormann, "Semantic Definition Format (SDF) for Data and Interactions of Things," Working Draft, Internet Engineering Task Force, Internet-Draft, 1 2022.

[25] J. Kiesewalter and C. Bormann, "Mapping between YANG and SDF," Working Draft, Internet Engineering Task Force, Internet-Draft, 11 2021.

[26] K. Sahlmann, "Network management with semantic descriptions for interoperability on the Internet of Things," PhD dissertation, University of Potsdam, 2021. [Online]. Available: https://doi.org/10.25932/publishup-52984

[27] Texas Instruments, "CC2538 System-on-Chip Solution for 2.4-GHz IEEE 802.15.4 and ZigBee/ZigBee IP Applications (Rev. C): User's Guide," Tech. Rep., 2012-2013.

[28] K. Sahlmann, T. Scheffler, and B. Schnor, "Managing IoT device capabilities based on oneM2M ontology descriptions," in *Proceedings of the 16. GI/ITG KuVS Fachgespräch Sensornetze*. HAW Hamburg, 2017, pp. 23–26.

[29] "Base Ontology: oneM2M Technical Specification," Tech. Rep. TS-0012-V3.7.1, 3 2018.

[30] K. Sahlmann, V. Clemens, M. Nowak, and B. Schnor, "MUP: Simplifying Secure Over-The-Air Update with MQTT for Constrained IoT Devices," *Sensors*, vol. 21, no. 1, 2021.

[31] M. Nowak, "Einbindung von Sensoren und Aktuatoren in die prototypische Implementierung der Verwaltung von IoT Geräten auf Basis von oneM2M Ontologie Beschreibungen (in German)," University of Potsdam, Institute of Computer Science, Project Study, 2018.

[32] C. Bormann, M. Ersue, and A. Keranen, "Terminology for Constrained-Node Networks," Internet Requests for Comments, Internet Engineering Task Force, RFC 7228, 5 2014.

[33] J. Ma, X. Zhou, S. Li, and Z. Li, "Connecting Agriculture to the Internet of Things through Sensor Networks," in *International Conference on Internet of Things and 4th International Conference on Cyber, Physical and Social Computing*, 2011, pp. 184–187.

[34] J. Bauer and N. Aschenbruck, "Measuring and Adapting MQTT in Cellular Networks for Collaborative Smart Farming," in *2017 IEEE 42nd Conference on Local Computer Networks (LCN)*, 2017, pp. 294–302.

[35] I. Marcu, C. Voicu, A. M. C. Dragulinescu, O. Fratu, G. Suciu, C. Balaceanu, and M. M. Andronache, "Overview of IoT basic platforms for precision agriculture," in *International Conference on Future Access Enablers of Ubiquitous and Intelligent Infrastructures*, 2019, pp. 124–137.

[36] M. E. H. Chowdhury, A. Khandakar, S. Ahmed, F. Al-Khuzaei, J. Hamdalla, F. Haque, M. B. I. Reaz, A. Al Shafei, and N. Al-Emadi, "Design, Construction and Testing of IoT Based Automated Indoor Vertical Hydroponics Farming Test-Bed in Qatar," *Sensors*, vol. 20, no. 19, 2020.

[37] H. Rijgersberg, D. Willems, X.-Y. Ren, M. Wigham, and J. Top, "Ontology of units of measure (om) 2.0," eFoodLab, Tech. Rep., 12 2017.

[38] S. Cox and C. Little, "Time ontology in OWL," W3C, Candidate Recommendation, 3 2020. [Online]. Available: https://www.w3.org/TR/2020/CR-owl-time-20200326/

[39] R. Jain, *The art of computer systems performance analysis: Techniques for experimental design, measurement, simulation, and modeling*. John Wiley & Sons Inc, 1991.

[40] RDFLib Library. [Online]. Available: https://github.com/RDFLib/rdflib

[41] K. Sahlmann, A. Lindemann, and B. Schnor, "Binary Representation of Device Descriptions: CBOR versus RDF HDT," in *Proceedings of the 17. GI/ITG KuVS Fachgespräch Sensornetze: Drahtlose Sensornetze*, 2018.

[42] J. D. Fernández, M. A. Martínez-Prieto, C. Gutierrez, and A. Polleres, "Binary RDF Representation for Publication and Exchange (HDT)," W3C Member Submission, 2011. [Online]. Available: https://www.w3.org/Submission/HDT/

[43] C. Bormann and P. Hoffman, "Concise Binary Object Representation (CBOR)," Internet Requests for Comments, Internet Engineering Task Force, RFC 7049, 10 2013.

[44] A. Bierman, M. Bjorklund, and K. Watsen, "RESTCONF Protocol," Internet Requests for Comments, Internet Engineering Task Force, RFC 8040, 1 2017.