Combining the Concepts of Semantic Data Integration and Edge Computing

Matthias Farnbauer-Schmidt[1,2], Julian Lindner[3], Christopher Kaffenberger[4], Jens Albrecht[5]

Abstract: The *Internet of Things* (IoT) is growing rapidly. Therefore, there are more and more vendors, which led to IoT being a heterogeneous collection of different IoT platforms, isolated solutions and several protocols. It has been proposed to use Data Integration to overcome this heterogeneity. In addition, costs are on the raise due to increasing volume of data which increases demands on bandwidth and cloud computing capabilities. Again a solution has already been proposed by reducing the amount of data to forward by processing data at the edge of an IoT-System, e. g. filtering or aggregation. This concept is called Edge Computing.

In this article the Semantic Edge Computing Runtime (SECR) is introduced, combining both concepts. The application of Data Integration enables Edge Computing to be performed on a higher level of abstraction. In addition, the developed Driver-approach allows SECR's Data Integration algorithm to be applied to a wide range of data sources without imposing requirements on them. The Data Integration itself is based on technologies of Semantic Web, applying metadata to raw data giving it context for interpretation. Furthermore, SECR's REST-API enables applications to alternate Data Integration and Edge Computing at runtime.

The tests of SECR's prototype implementation have shown its suitability for deployment on an edge device and its scalability, being able to handle 128 data sources and Edge Computing Tasks.

Keywords: Internet of Things; Data Integration; Edge Computing; Semantic Web; SECR

1 Introduction

The *Internet of Things* (IoT) is the approach of linking the real world to the Internet. Consequently, digitalization of real-world properties is done by measurements conducted by sensors.

The dominant architecture of IoT applications relies on a central cloud, i. e. a powerful computational center. All data produced by sensors is forwarded to the cloud for processing,

[1] Technische Hochschule Nürnberg Georg-Simon-Ohm, Keßlerplatz 12, 90489 Nürnberg, Germany
[2] Fraunhofer IIS Arbeitsgruppe SCS, Nordostpark 93, 90411 Nürnberg, Germany farnbams@scs.fraunhofer.de
[3] Fraunhofer IIS Arbeitsgruppe SCS, Nordostpark 93, 90411 Nürnberg, Germany julian.lindner@scs.fraunhofer.de
[4] Fraunhofer IIS Arbeitsgruppe SCS, Nordostpark 93, 90411 Nürnberg, Germany christopher.kaffenberger@scs.fraunhofer.de
[5] Technische Hochschule Nürnberg Georg-Simon-Ohm, Informatik, Keßlerplatz 12, 90489 Nürnberg, Germany jens.albrecht@th-nuernberg.de

doi:10.18420/inf2019_19

storing and decision making. Furthermore, there are Gateways that translate protocols on the way from data source to cloud.

Scalability problems of the cloud-centric IoT-architecture are pointed out by the growing number of devices [Ga17]. As a result, the more devices are deployed the more data is produced. With an increased volume of data, a cloud requires higher computational resources. Moreover, the network connecting devices and cloud must provide a higher bandwidth to be able to convey it. In fact, bandwidth is a constraint resource and both, computational power and bandwidth are expensive. A solution to this problem is introduced by Edge Computing where data is pre-processed at the edge.

The IoT is highly heterogeneous today [Qi18]. It can be seen at every layer of the ISO-OSI-model. In addition, the representation of data within a protocol can be heterogeneous either. For instance, there can be differences in units, scale and meaning. In fact, temperature of 32 could mean 32 m°Cor 32 K and could be the room temperature or the average temperature in space. Representation is usually defined by contract at protocol, platform or application level. Besides, some domains have their own niche solutions. A proper way to overcome heterogeneity of different data sources is to perform Data Integration. The approach of using Semantic Web Technology has been introduced to the IoT and is called Semantic Web of Things.

Edge Computing requires Data Integration when computations should be applied to data from different sources. In order to produce sensible results, computations require their inputs to be modeled according to the same schema. In fact, the Data Integration decouples the execution of computations from the heterogeneity of data sources. As a result, Edge Computing software that includes a Data Integration layer is more reusable than Edge Computing software that handles specific data sources.

The benefits of combining Semantic Data Integration and Edge Computing will be shown by introducing the Semantic Edge Computing Runtime (SECR). Focused on performing pre-processing for data science algorithms, it works as a backend for IoT-applications on the edge. SECR is designed to be deployed at edge devices at least capable of running an OS. This excludes the outermost edge devices like simple sensors and actuators. An abstraction of data sources in combination with the developed Driver-approach allows SECR's Semantic Data Integration to handle a wide range of data sources. In addition, a local RDF-graph is maintained that provides all information of SECR, its host and environment. It is internally used for configuration of services, either. Furthermore, a REST-API is provided to access the graph. Moreover, the API allows for modification of Edge Computing tasks and Semantic Data Integration at runtime.

2 Background

This section addresses the solutions to the problems of IoT before mentioned. In addition, their background and technologies are covered.

2.1 Data Integration

Heterogeneity of data sources can be overcome by applying Data Integration. It is done by transforming data into a common schema. As a consequence, all integrated data can be queried as a whole. A schema is a description of how certain information is modeled. Although, a schema only defines the semantics of a data model not the syntax the data is represented in.

Data Integration enables interoperability if the communicators understand the common schema. The lower the system-layer Data Integration is applied the earlier interoperability between IoT-systems can be achieved.

2.2 Semantic Web

The Semantic Web or Web of Data wants to interlink the data provided in the Internet. This concept is called Linked Data [LPL17].

The standard used for Linked Data is the Resource Description Framework[6] (RDF) a recommendation of the World Wide Web Consortium (W3C). The Framework sees the description of information in subject-predicate-object-triples, e. g. “Hans is male”. Subjects and predicates must be resources identified by an Uniform Resource Identifier (URI) whereas objects can be either a resource or a literal. Several interlinked RDF-triples build a directed graph where subjects and objects are the nodes and the predicates are the directed edges.

2.2.1 Ontologies

An ontology describes entities and the relations between them. In case of the Semantic Web an ontology is defined by RDF-statements (RDF-triples). These statements are divided into two groups the terminological box (TBox) and the assertion box (ABox) [Bo17]. The TBox-statements provide classes and predicates to identify entities and their kind of relations. In contrast, the ABox-statements use the terms defined by the TBox to describe entities and their relations.

Depending on the share of TBox- and ABox-statements, ontologies are either classified as vocabulary or as knowledge-graph in this paper. This is done in order to express the purpose of an RDF-graph.

The TBox-statements of a vocabulary define a schema for modeling data. They can describe a broad domain or extend such a vocabulary into more detail. An example is the Semantic Sensor Network (SSN) ontology[7] that exapnds the Sensor, Observation, Sample and Actuator

[6] https://www.w3.org/RDF/
[7] https://www.w3.org/TR/vocab-ssn/

(SOSA) ontology. Both are ontologies provided by the W3C. The term ontology is often used as a synonym for vocabulary.

Knowledge-graphs use vocabularies to model entities and their relations. By using a common vocabulary the semantics of a graph can be understood by everyone that knows the vocabulary.

2.2.2 Data Integration by Application of Vocabularies

Building knowledge graphs by using the terms of vocabularies is a kind of Data Integration, in the future referred to as Semantic Data Integration (SDI). The schema built from a vocabulary's statements works as a common schema for Data Integration. Being a directed graph, the linked statements of an RDF-ontology can be traversed. Therefore, if a reader knows the vocabulary used to describe the entities of a graph he is able to infer the semantics of that graph.

2.3 Edge Computing

Edge Computing tackles IoT's issue of an increasing volume of data. It utilizes the execution of computations on edge devices. The concept leverages the computational powers of devices of the outer ends of an IoT-system to reduce the payloads for network and cloud.

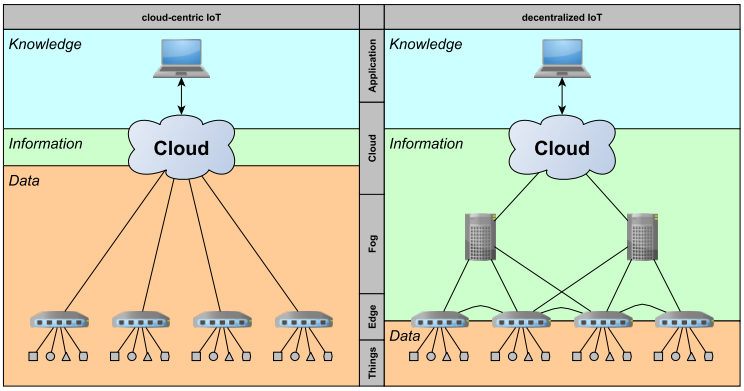

Fig. 1: Cloud-centric IoT-architecture (left) and decentralized IoT-architecture including Edge Computing (right). Conversion from Data to Information takes place at gateway level. Redrawn according to [Pa17].

The term Edge Computing is not globally defined. The definition of the "edge" used in this article is shown in Figure 1. It is composed of the gateways connecting sensors and actuators to the infrastructure of the IoT. Therefore, the Edge Computing introduced by

SECR only targets those edge devices. For example, Brown, Kathrivel and Akthar define Edge Computing to be executed on decentral micro-clouds [Br17; KA17] (see Fog in Figure 1). Whereas others see the edge as the outermost sensors and actuators (see Things in Figure 1).

The distribution of computations increases the complexity of an IoT-system because the cloud is no longer the only instance that conducts computations. Edge devices are heterogeneous and provide different levels of computational powers. As a consequence, a system has to be introduced to manage deployment of Edge Computing and load balancing. Such systems are referred to as Edge Computing platforms. In contrast, the software deployed at an edge device to execute Edge Computing is called Edge Computing software. An Edge Computing platform distributes tasks requested by an application to edge devices deploying Edge Computing software. After all, depending on the complexity of a system and the number of edge devices Edge Computing platforms are optional.

3 Related Work

Semantic Data Integration and data pre-processing has been suggested by other projects before and will be examined in the following.

Desai et al. introduced the concept of Semantic Gateway as a Service [Al15]. The purpose of it is to break up *vertical silos*. These are closed IoT-applications which are obstacles on the way to enable gateway-level interoperability. In fact, the concept is limited to Semantic Data Integration. Applications can access the results either by push-pull REST-API or by event-driven MQTT. Besides, a multi-protocol proxy is used for handling of data sources. In a Semantic Gateway the data sources must implement a certain protocol. So, the sources itself provide descriptions of their packages for the gateway. These descriptions are used to extract the contained data of a package. All in all, the requirement for data sources to implement a protocol limits the data sources that can be handled by the Semantic Gateway.

Semantic enrichment of data causes an increase of payload due to additional metadata. Al-Osta et al. further developed the concept of Semantic Gateway to reduce the data that must be forwarded by pre-processing incoming data [AAA17]. According to this report's definition this is Edge Computing. However, data sources are still required to implement a certain protocol to work within this system. Their Data Preparation Module reduces traffic by applying rules of aggregation and filtering. Consequently, Edge Computing capabilities are restricted. In contrast, SECR provides richer Edge Computing capabilities, allowing for dependencies between sources, scaling and converting of data.

To sum up, Semantic Gateway and its derivatives show that Semantic Data Integration can be done at gateway-level. In addition, Semantic Data Integration and rule-based data processing can not only reduce the emitted information but also create new information.

4 The Semantic Edge Computing Runtime

Designed as a backend for IoT-applications on the edge, SECR is Edge Computing software. Besides, the small footprint leaves enough resources to run further services on the same host. All results of SECR's services are published as RDF-graphs enabling edge-level interoperability. Moreover, SECR's Edge Computing capabilities focus on data processing, e. g. filtering, aggregation, fuzzyfication and classification.

Publish-subscribe HTTP and event-driven MQTT is used for SECR's public API (see Figure 2). The HTTP-API is used to pull results from SECR's services, configure the services and for querying SECR's local RDF-graph whereas MQTT is used for event-driven publication of service results. In MQTT content is published to so-called topics. The protocol is handled by a broker, which notifies all subscribers of a topic when new content is published. The Data Source Managers (DSMs) in Figure 2 are a proxy for the instances that handle the Semantic Data Integration. Similarly, the Edge Computing Tasks (ECTs) do the Edge Computing.

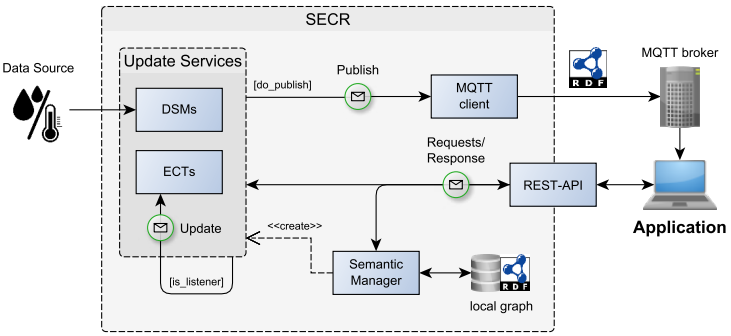

Fig. 2: Architecture of SECR. Applications can either interact with the REST-API or the MQTT-broker. Update Services are launched by the Semantic Manager which holds access to SECR's local RDF-graph. Edge Computing relies on the results of other Update Services.

The components of SECR will be further discussed and explained in the rest of this section.

4.1 SECR's Public Services

Metadata of SECR can be obtained from the public accessible local RDF-graph and from the Resource-Usage-Information service. Provided are memory allocation and CPU load of SECR's host system, the average latencies of Edge Computing and Semantic Data Integration as well as a description of the host system, its environment and SECR's deployed services. For reasons of clarity these services are not shown in Figure 2.

A vital part of SECR are Update Services. Each Update Service handles a Semantic Struct which is SECR's representation of results from Semantic Data Integration or Edge

Computing. An Update Service's purpose is to provide updates of a Semantic Struct to consumers of the service's results. Furthermore, Semantic Structs can be serialized into an RDF-graph. A user can subscribe to a Semantic Struct's state at an MQTT-topic. However, an Update Service must be set to publish to MQTT.

4.2 Semantic Structs

Semantic Structs consist of a timestamp of their last update, the URI of their Update Service and at least one Field identified by a label, an URI and the type of data they hold.

```
@prefix rdf  <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix sosa <http://www.w3.org/ns/sosa/> .
@prefix sec  <http://iis.fraunhofer.de/vocab/sec/> .

_:obs a sosa:Observation ;
  sosa:usedProcedure {Update-Service-URI} ;
  # for each of Semantic Struct's Fields
  sosa:hasResult [
        sec:instanceOF {Field-URI} ;
        rdf:value {current-value-of-Field}
  ] ;
  sosa:resultTime {Semantic-Structs-timestamp as
       YYYY-MM-DDThh:mm:ss.sss} .
```

Fig. 3: Pattern of an RDF-graph representing the state of a Semantic Struct. The Turtle-syntax is used.

The state of a Semantic Struct is returned as an RDF-graph to users. The RDF-graph resembles a `sosa:Observation` following the pattern of Figure 3. The chosen graph-pattern provides all information necessary to discover the full semantic description of the Semantic Struct and the changing values of Fields and time of update.

4.3 Semantic Data Integration

Semantic Data Integration is performed by a combination of Drivers, Data Source Managers (DSMs) and Semantic Conversion Services (SCSs). The components of SECR's Semantic Data Integration layer are shown in Figure 4 and described in the following paragraphs.

Data Sources are an abstraction used by SECR's Semantic Data Integration to handle IoT's heterogeneous data sources. A *Data Source* is able to emit packages of raw data. These packages are called Frames which consist of different Fields similar to Fields of a Semantic Struct. The Fields of a Frame hold the raw data.

For each *Data Source* SECR handles, a full semantic description is provided in SECR's local RDF-graph. Different *Data Sources* can be of the same type, e. g. several sensors (entities)

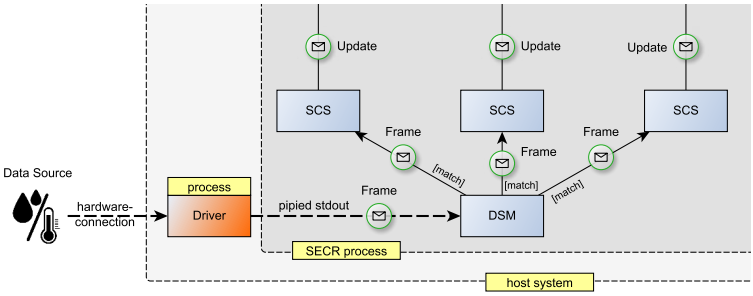

Fig. 4: SECR's Semantic Data Integration of a *Data Source* is a collaboration of several components.

of the same model (type) could be deployed. A type's description defines the Driver to be used and which Frames are emitted and their properties. Furthermore, a Frame's description includes its URI, a label, a pattern that uniquely identifies the Frame, the description of its Fields and information how to extract the data of the Fields. Accepted Frame formats are byte-arrays, ASCII-strings, XML and Json. A Frames pattern depends on its format. For example, the pattern of a byte-array Frame is a sequence of bytes that must match the Frame's content. In contrast, the pattern of a Json-format Frame is a key-value-pair that must be contained in the Json. As of Semantic Structs, Fields of a Frame are defined by an URI, a label and their type of data. The encoding of the Fields as well as the pattern depends on a Frame's format. For example for byte-array Frames a Field is localized by position and length in the array whereas Json- and XML-encodings provide the keys where the data is found.

The developed **Driver**-approach allows SECRs to abstract over IoT's heterogeneous data sources. A Driver's purpose is to validate a *Data Source*'s protocol, to handle the connection between the *Data Source* and the SECR and to forward validated Frames to its DSM. For each *Data Source* handled by SECR one Driver is launched. As depicted in Figure 4 a Driver is a child-process launched by SECR. Therefore, Drivers can be configured by passing command-line-arguments at their launch. Arguments can be specified for each *Data Source type*, for each *Data Source* and at setup of a DSM.

As an example, we assume a HTTP-server as *Data Source*. Therefore, the Driver would be responsible to continuously request new Frames from the server. In addition, the response bodies contain a custom checksum that must be validated by the Driver to proof a Frames validity.

A **Data Source Manager** (DSM) coordinates the conversion of Frames into Semantic Structs. This is done by receiving validated Frames from the Driver and passing them to the responsible Semantic Conversion Service. Therefore, it is the DSM's task to determine which kind of Frame is received by applying the patterns of the handled *Data Source*'s Frames.

For each Frame a *Data Source* can emit a **Semantic Conversion Service** is deployed by the Data Source Manager. They are Update Services whose task is to convert a Frame into their Semantic Struct. The conversion is done by extracting the raw data of the Frame and updating the corresponding Fields of the SCS's Semantic Struce. For extraction the encoding information from the Frame's description is used.

4.4 Edge Computing Tasks

Edge Computing Tasks (ECTs) are SECR's source of Edge Computing capabilities. Consequently, each ECT is an Update Service. Indeed, the Edge Computing is done by calculating new values for the Fields of an ECT's Semantic Struct. For each Field an expression is provided. SECR's supported types of data are *Numeric*, *Boolean* and *Categorical*. Furthermore, a Boolean expression called `publish-condition` is provided for each ECT. It defines the moments when an ECT updates the state of its Semantic Struct.

4.4.1 Setup of an ECT

The creation of an ECT requires a SECR-wide unique label, description of the new Semantic Struct's Fields, the expression for them and the `publish-condition`. Accordingly, all required information must be provided except for the ECT's label which is encoded in the HTTP-request's URL.

For example we want to create a new ECT called `task` on a SECR (`${base} = http://example.org:8080/secr0`). On the SECR a SCS (`${base}/dsm/other/scs/frame`) and an ECT (`${base}/ect/another`) are already running. Field `x` of the new ECT should be calculated from the value of SCS's Field `a` and 30 and Field `y` should be calculated from the value of `x` and value `t` of the other ECT. At last, updates should be published when `x` exceeds 50. We achieve the described behaviour by sending the Json-LD from Figure 5 to `POST http://example.org:8080/secr0/ect/task`.

Unlike most of the required semantics, the expressions are not encoded as RDF (see Figure 5). This decision was made for user friendliness because complex expressions would result in large and complex RDF-graphs.

Fields in expressions are accessed by `{identifier}#{FieldLabel}` where an identifier is either the URI of another Update Service or `SELF` which indicates that the Field is part of the ECT's own Semantic Struct. Own Fields can only be referenced when they have been declared before, e. g. Field `x` could not reference Field `y`.

```
// replace ${base} by http://example.org:8080/secr0

{ "@context":"${base}/context.json",
  "dependencies": [
        { "other":"${base}/dsm/other/scs/frame" },
        { "another":"${base}/ect/another" }
  ],
  "fields": [ {
          "label":"x", "ofType":"Numeric",
          "expression":"<other#a> + 30"
        }, {
          "label":"y", "ofType":"Numeric",
          "expression":"<another#t> + SELF#x"
        } ],
  "publish_when": { "expression":"<SELF#x> > 50" }
}
```

Fig. 5: Example content of POST to create a new Edge Computing Task. The Semantic Struct will consist of Fields `x` and `y`. The ECT will depend on SCS `other/frame` and ECT `another`.

4.4.2 Algorithm of Evaluation

The evaluation of ECT's expressions is driven by the updates of the services they depend on. For each dependency an execution plan is created, e. g. the execution plan for the example ECT `task` is shown in Table 1. Indeed, the Update Service `other` is only mentioned in the expression for Field `x` (see Figure 5). However, the execution plan for updates from `other` additionally recalculates Field `y` and the `publish-condition` because they depend on Field `x`.

Tab. 1: Resulting execution plans from the instruction of Figure 5.

Update from	**Execution plan**
other	Recalculate `x` → recalculate `y` → recalculate `publish-condition`
another	Recalculate `y`

5 Evaluation

The prototype implementation of SECR is tested for scalability and suitability for deployment at the edge.

For evaluation purposes, a special Driver has been implemented. The Driver itself simulates a *Data Source* that emits every 50 ms one Frame. Furthermore, the *Data Source* can send three different kinds of Frames which one is sent is determined by chance.

5.1 The Test Scenario

For the tests SECR is deployed on a RaspberryPi 3 Model B that runs a quad-core Arm-processor at 1.2 GHz.

For testing the scalability several tests are run with 2 up to 128 (2, 4, 8, 16, 32, 64, 96, 128) simulated *Data Sources* at one time. All tests are run for 120 s 30-times. Four test cases have been evaluated:

1. n DSMs are deployed; Nothing is published to MQTT.
2. n DSMs are deployed; All results are published to MQTT.
3. n DSMs and $n - 4$ ECTs are deployed; Nothing is published to MQTT.
4. n DSMs and $n - 4$ ECTs are deployed; All results are published to MQTT.

In the third and fourth case the ECTs depending on four SCSs, created from instructions like the example in Figure 6. Indeed, the fourth test case can be seen as a worst-case scenario.

```
// replace ${base} by http://localhost:8080/secr0

{ "@context":"${base}/context.json",
  "dependencies": [
        { "s31_temp":"${base}/dsm/S31/scs/temp" },
        {  "s32_vel":"${base}/dsm/S32/scs/vel" },
        { "s33_temp":"${base}/dsm/S33/scs/temp" },
        { "s34_temp":"${base}/dsm/S34/scs/temp" }
  ],
  "fields": [ {
          "label":"temp_avg", "ofType":"Numeric",
          "expression":"mov_avg(4, s31_temp#t * 0.01)"
        }, {
          "label":"mul", "ofType":"Numeric",
          "expression":"s32_vel#v * s33_temp#t"
        } ],
  "publish_when": { "expression":"s34_temp#t.ROSE" }
}
```

Fig. 6: Body of `POST http://localhost:8080/secr0/ect/alert34`. Instruction to create ECT `alert34` for the evaluation of SECR's prototype implementation.

Time and resource consumption measurements are part of SECR's Resource-Usage-Information service. Therefore, the measurements do not generate any extra costs.

The latencies of the Update Services where measured to determine the payload possible to be handled by SECR running on a RaspberryPi 3. On the one hand, the latency of Semantic Data Integration defines the time elapsed from the moment a new Frame is read from the

Driver's `stdout` to the moment the serialized RDF is sent to the MQTT-broker. On the other hand, the latency of Edge Computing is defined as the time elapsed from the moment the ECT received the notification of update to the moment the serialized RDF is sent to the MQTT-broker.

5.2 Results and Discussion

The results of the tests are shown in Figure 7 through Figure 10. Note the non-linear x-axis. In addition, it should be considered that due to other processes the latencies can be disturbed. Therefore, the results are presented as boxplots.

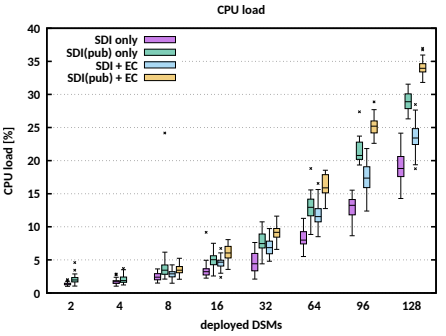

Fig. 7: CPU load of the RaspberryPi 3 when running SECR.

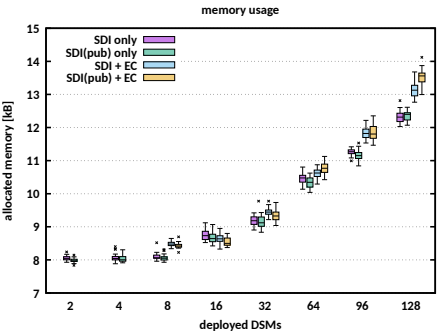

Fig. 8: Allocated memory of SECR's process on the RaspberryPi 3.

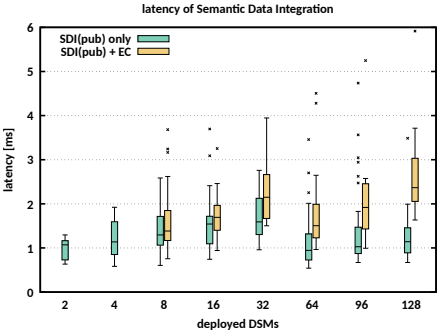

Fig. 9: Latency of Semantic Data Integration.

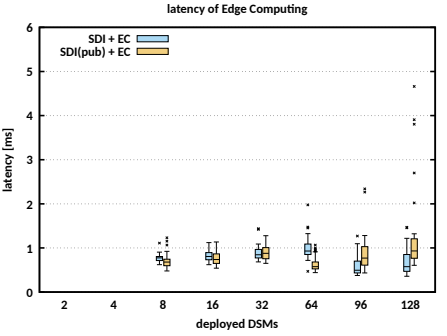

Fig. 10: Latency of Edge Computing.

As expected, in each test case the CPU load rises nearly linear with the number of *Data Sources* handled. By capturing one third of the hosts CPU in a worst-case scenario, SECR leaves enough resources to run further processes achieving the objective defined in section 4.

The memory allocation of SECR's process starts with an offset of 8 kB and rises in all test cases nearly linear. The offset is mainly derived from SECR's local RDF-graph. By

allocating 13.5 kB in test case 4, SECR is suitable to be deployed at systems with limited memory capacities.

The latency of Semantic Data Integration shows a number of outliers but stays generally below 4 ms. The latencies of both Semantic Data Integration and Edge Computing rise with the number of deployed services due to emerging dependencies between them. In the worst-case a new Frame is integrated within 6 ms and processed by an ECT within 5 ms. Finally, the sum of both propagation times is nearly 5-times faster than the occurrence of new Frames.

6 Conclusion and Future Work

In this paper the authors introduced the Semantic Edge Computing Runtime (SECR). The software enables Edge Computing capabilities on the host system. The heterogeneity of the Internet of Things (IoT) is handled by applying Data Integration before the Edge Computing. Besides, SECR's Data Integration is done by converting raw data into RDF-graphs. This abstraction before the computations allows SECR to apply Edge Computing on a wide range of *Data Sources*.

The evaluation of the prototype SECR has proven its suitability for deployment at the edge. Compared to the results of [AAA17] the overall worst latency of 11 ms is 3-times faster than their average latency of 30 ms.

In the future we will examine a real-world use-case to investigate the effects of Edge Computing in terms of saving bandwidth. In order to make SECR production ready concerns are taken towards security and failure safety.

With more computations taking place at the edge the interest of attackers raises. Therefore, measures must be taken to prevent malicious attacks. Nevertheless, edge devices are resource constraint. So, a compromise must be found between resource consumption and safety measures.

The loss of functionality after failure must be prevented. Currently, SECR provides no persistence of its services. Certainly, edge devices are more prone to failures than computational centers. In addition, it is advantageous to turn of edge devices to save energy sometimes. On restart SECR should restore the state of its services.

7 Acknowledgement

This work was partially supported by the Bavarian State Ministry of Economic Affairs, Regional Development and Energy within the framework of the Bavarian Research and Development Program "Information and Communication Technology".

References

[AAA17] Al-Osta, M.; Ahmed, B.; Abdelouahed, G.: A Lightweight Semantic Web-based Approach for Data Annotation on IoT Gateways. International Conference on Emerging Ubiquitous Systems and Pervasive Networks 8th/, 2017.

[Al15] Semantic Gateway as a Service Architecture for IoT Interoperability. In (Altintas, O., ed.): 2015 IEEE International Conference on Mobile Services (MS). IEEE, Piscataway, NJ, pp. 313–319, 2015, ISBN: 978-1-4673-7284-8.

[Bo17] Bonte, P.; Ongenae, F.; Backere, F.; Schaballie, J.; Arndt, D.; Verstichel, S.; Mannens, E.; Walle, R.; Turck, F.: The MASSIF Platform: A Modular and Semantic Platform for the Development of Flexible IoT Services. Knowl. Inf. Syst. 51/1, pp. 89–126, 2017, ISSN: 0219-1377, URL: https://doi.org/10.1007/s10115-016-0969-1.

[Br17] Brown, K.: Resiliency of Edge Data Centers in the Era of Cloud Computing, YouTube, 2017, URL: https://www.youtube.com/watch?v=ttto-t4asE0, visited on: 07/31/2018.

[Ga17] Gartner, I.: Gartner Says 8.4 Billion Connected Things Will Be in Use in 2017, Up 31 Percent From 2016, 2017, URL: https://www.gartner.com/en/newsroom/press-releases/2017-02-07-gartner-says-8-billion-connected-things-will-be-in-use-in-2017-up-31-percent-from-2016, visited on: 11/07/2018.

[KA17] Kathirvel, K.; Akhtar, H.: Implications of 5G and Edge Computing on OpenStack, Youtube, 2017, URL: https://www.youtube.com/watch?v=9d5JtONGQSA, visited on: 07/31/2018.

[Ka18] Kaed, C. E.; Khan, I.; van den Berg, A.; Hossayni, H.; Saint-Marcel, C.: SRE: Semantic Rules Engine for the Industrial Internet-Of-Things Gateways. IEEE Transactions on Industrial Informatics 14/2, pp. 715–724, 2018, ISSN: 1551-3203.

[LPL17] Li, W.; Privat, G.; Le Gall, F.: Towards a Semantics Extractor for Interoperability of IoT Platforms. Global Internet of Things Sumit/, 2017.

[Pa17] Pande, A.: IOT Edge Computing | IoT Examples | Use Cases | HackerEarth Webinar, YouTube, 2017, URL: https://www.youtube.com/watch?v=Xm8frqTZRVI, visited on: 07/31/2018.

[Qi18] Qiu, T.; Chen, N.; Li, K.; Atiquzzaman, M.; Zhao, W.: How Can Heterogeneous Internet of Things Build our Future: A Survey. IEEE Communications Surveys & Tutorials/, p. 1, 2018.