Towards Cooperative Semantic Computing: A Distributed Reasoning Approach for Fog-Enabled SWoT

Nicolas Seydoux[1,2(✉)], Khalil Drira[1], Nathalie Hernandez[2], and Thierry Monteil[1]

[1] LAAS-CNRS, Universit de Toulouse, CNRS, INSA, Toulouse, France
{nicolas.seydoux,khalil.drira,thierry.monteil}@laas.fr
[2] IRIT, Maison de la Recherche, Univ. Toulouse Jean Jaurs, 5 alles Antonio Machado, 31000 Toulouse, France
nathalie.hernandez@irit.fr

Abstract. The development of the Semantic Web of Things (SWoT) is challenged by the nature of IoT architectures where constrained devices are connected to powerful cloud servers in charge of processing remotely collected data. Such an architectural pattern introduces multiple bottlenecks constituting a hurdle for scalability, and degrades the QoS parameters such as response time. This hinders the development of a number of critical and time-sensitive applications. As an alternative to this Cloud-centric architecture, Fog-enabled architectures can be considered to take advantage of the myriad of devices that can be used for partially processing data circulating between the local sensors and the remote Cloud servers. The approach developed in this paper is a contribution in this direction: it aims to enable rule-based processing to be deployed closer to data sources, in order to foster the implementation of semantic-enabled applications. For this purpose, we define a dynamic deployment technique for rule-based semantic reasoning on Fog nodes. This technique has been evaluated according to a strategy improving information delivery delay to applications. The implementation in Java based on SHACL rules has been executed on a platform containing a server, a laptop and a Raspberry Pi, and is evaluated on a smart building use case where both distribution and scalability have been considered.

Keywords: Distributed reasoning · Cloud-Fog processing
Semantic Web of Things · Cooperative semantic computing
SHACL rules

1 Introduction

The maturity of Internet of Things (IoT) communication technologies is fostering a wide variety of industrial and societal applications with responsiveness and privacy requirements, especially useful for home automation or industry

H. Panetto et al. (Eds.): OTM 2018 Conferences, LNCS 11229, pp. 407–425, 2018.
https://doi.org/10.1007/978-3-030-02610-3_23

4.0 scenarios. However, the development of IoT service platforms and related applications faces new challenges emerging from data heterogeneity on the one hand and real-time decision management on the other. Such challenges can be addressed by investigating **semantic interoperability** and **automated reasoning** techniques that respectively allow IoT systems to exchange data with a shared understanding, and to infer new information from existing data and formalized knowledge. These requirements, crucial to IoT applications, are fostering the adoption of the Semantic Web (SW) principles and technologies as interoperability enablers [9]. A new domain has emerged from the interaction between the IOT and the SW, the Semantic Web of Things (SWoT) [15]. The number of connected devices [21] and the volume of generated data to be processed by SWoT systems are growing substantially, requiring **scalability** to be addressed as an additional requirement. Moreover, SW technologies are resource-consuming, and go beyond the capacities of constrained IoT devices, leading to centralizing processing on remote powerful nodes. The SW stack components are often executed on the Cloud, where servers collect and process IoT data before feeding applications with curated data, analytics results and decision instructions [16]. In such a centralized architecture, the Cloud becomes a single point of failure, a bottleneck for communication and a threat for privacy. Storing and processing a large data volume in a central place induces delay [21] and degrades quality of service for IoT applications. It may hamper the development of a variety of applications and inhibit the implementation of time-critical applications. Our objective is to face these issues by applying the Fog computing paradigm to the SWoT infrastructure by considering devices located between constrained sensors and the Cloud as initiated in [13]. Even if constrained, Fog nodes also provide computation capabilities that are starting to be used to process data closer to the sources it is generated from [11,13]. These limited computation capabilities are also leveraged by the development of dedicated SW libraries, such as μJena[1]. The purpose here is not to replace the Cloud by the Fog, but to **complement their capabilities** to overcome their respective limits, thus achieving **Cloud-Fog cooperative semantic computing**.

IoT applications, such as an automatic light manager or a smart city traffic monitor, exploit data collected by device networks for ad-hoc purposes. These applications usually consume IoT data from Cloud nodes, processed to provide high-level information relevant to the application end user [2]. Rules can be used to capture the specific needs of applications by representing and sharing dedicated deduction intent [19]. Existing IoT architectures mainly process rules on the Cloud [4]. In this paper we propose an approach in which Fog nodes **produce information of concern for applications directly** by applying rules on sensor observations, instead of simply forwarding raw data to remote Cloud nodes. Moreover, the diversity of IoT applications implies **multiple strategies for rule deployment**: applications may favor response time, privacy, or energy consumption, and rules should be deployed in the Fog accordingly. For instance, rules applied on nodes closer to sensors can yield deductions faster compared to

[1] http://poseidon.ws.dei.polimi.it/ca/?page_id=59.

a centralized approach, both because they receive data earlier and because they process a smaller amount of observations, reducing the end-to-end reasoning time [11,22]. Identifying such Fog nodes is a possible rule deployment strategy. However, identifying which node should process which rule in order to provide applications with the required results while minimizing resource consumption is challenging, especially in a dynamic setting which characterizes IoT networks.

Our contribution aims at **distributing semantic data processing** in the Fog, according to **adaptable rule and data deployment strategies** defining an **optimal rule deployment** among nodes. We propose Emergent Distributed Reasoning (EDR), a **scalable**, strategy-agnostic approach to semantic processing based on **dynamic** rule deployment in a device network. The core approach is based on a deployment technique enabled by dedicated knowledge representations, and the various strategies are implemented based on modular rules expressed in SHACL, a recent W3C formalism. EDR relies on knowledge local to nodes: each node propagates reasoning rules toward the device network edge. In this paper, EDR is instantiated with a strategy aiming at **increasing application responsiveness**, EDR$_{\mathcal{PT}}$, based on **p**roperty **t**ypes of observations produced by sensors (e.g., temperature or luminosity).

The remainder of this paper is organized as such: Sect. 2 describes a smart factory use case. In Sect. 3, existing approaches are studied and compared to identify the innovations in EDR. The core contribution is detailed in Sect. 4, with the associated hypothesis and knowledge representations. EDR and EDR$_{\mathcal{PT}}$ are evaluated in Sect. 5, based on the smart factory use case. Finally, the paper is concluded in Sect. 6.

2 Motivating Use Case

Let us consider a production plant divided into two floors, processing different kind of products as illustrated in Fig. 1. These floors are modular: the structure described thereafter is subject to change to adapt to new productions. Each floor is equipped with conveyor belts carrying products from machine to machine for transformation. Devices are organized hierarchically: machines are connected to conveyors that are connected to the floor gateway, that collects and delivers data to the factory datacenter. The factory is equipped with sensors to ensure the safety of workers: each floor is equipped with presence, luminosity particle and temperature sensors, and the workers are equipped with wearables that automatically communicate in BLE[2] with nearby conveyors. Observations from the different sensors are used to identify potentially harmful situations, and then notify the control center, where actions can be taken remotely. Unsafe situations are described with deduction rules, based on the semantic description of observations and of the environment. Examples of rules include "the activation of a machine creating sparks in an atmosphere loaded with particles creates a detonation hazard", or "The presence of a worker near an operating machine in a low luminosity environment is a personal security hazard". Some rules are also

[2] https://en.wikipedia.org/wiki/Bluetooth_Low_Energy.

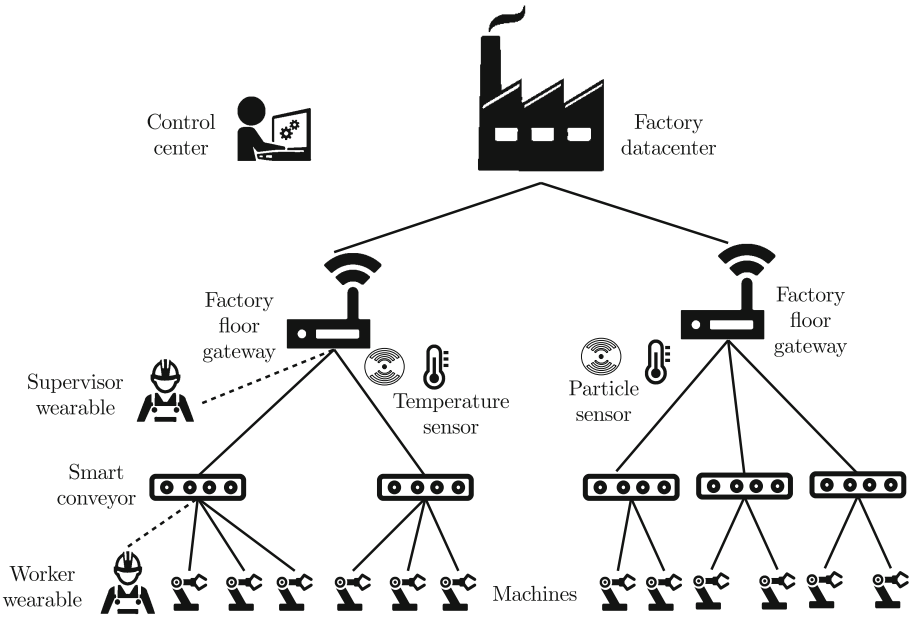

Fig. 1. Fog-enabled smart factory

dedicated to quality insurance: sensors available in the factory, such as temperature sensors, or sensors integrated to machines and to the conveyor, enable the continuous control of production quality. Some operations are temperature-sensitive, and a quality insurance rule is "The detection of a temperature above a certain threshold is a break in the cold chain". Adapting the speed of conveyors to the speed of machines is also part of quality enforcement. All the rules are summarized in Table 1, and their SHACL representation is available online[3].

Safety and quality insurance are time-sensitive applications, which is why the processing of the rules should be as fast as possible. Moreover, the mobility of some sensors (workers wearable), combined to the modularity of the factory floors, are suitable for a dynamic solution adaptative to their evolution over time.

3 Related Work

As the concern of the proposed approach is to deploy reasoning rules among Fog nodes to enable the deduction of application-dedicated information from IoT data, state-of-the-art work dealing with logical rules for the SW, logical rules for the IoT, distributed reasoning and processing on constrained nodes is presented.

Complementary to domain knowledge representation through ontologies, logical rules can be seen as a paradigm for knowledge modeling dedicated to specific

[3] https://w3id.org/laas-iot/edr/iiot/iiot.tar.gz.

Table 1. Safety and quality rules

Rule ID	Rule core
R1: Low machine visibility	$Location(l) \wedge Presence(l, o_1) \wedge o_1 = True \wedge Luminosity(l, o_2)$ $\wedge o_2 < 300L \wedge Machine(m) \wedge Activity(m, o_3)$ $\wedge o_3 = True \wedge locatedIn(m, l) \rightarrow LowMachineVisibility(m)$
R2: Low conveyor visibility	$Location(l) \wedge Presence(l, o_1) \wedge o_1 = True \wedge Luminosity(l, o_2)$ $\wedge o_2 < 300L \wedge Conveyor(c) \wedge Activity(c, o_3)$ $\wedge o_3 = True \wedge locatedIn(c, l) \rightarrow LowConveyorVisibility(c)$
R3: No supervision	$Location(l) \wedge Presence(l, o_1) \wedge o_1 = False$ $\wedge Conveyor(c) \wedge Activity(c, o_3) \wedge o_3 = True \wedge locatedIn(c, x)$ $\wedge SupervisorPost(s) \wedge supervises(s, c) \rightarrow NoSupervision(c)$
R4: Fire hazard	$Location(l) \wedge ParticleLevel(l, o_1) \wedge o_1 > 25\%$ $\wedge SparkMachine(m) \wedge Activity(m, o_3) \wedge o_3 = True$ $\wedge locatedIn(m, l) \rightarrow Firehazard(m)$
R5: Cold chain broken	$Location(l) \wedge Temperature(l, o_1) \wedge o_1 > 6^oC$ $\wedge TemperatureSensitiveMachine(m) \wedge Activity(m, o_3)$ $\wedge o_3 = True \wedge locatedIn(l, m) \rightarrow ColdChainBroken(m)$
R6: Conveyor too fast	$Conveyor(c) \wedge Machine(m) \wedge onConveyor(m, c)$ $\wedge MachineSpeed(m, s_m) \wedge ConveyorSpeed(c, s_c) \wedge s_c > s_m$ $\rightarrow ConveyorTooFast(c)$
R7: Low quality product	$Machine(m) \wedge ProductQuality(m, o_1) \wedge o_1 < 98.5$ $\rightarrow LowQualityProduct(m)$

usages. Logical rules are purely used for deduction: if their preconditions are true, the engine deduces their postconditions. With the goal of facilitating rule reuse, Linked Rules principles have been proposed [8]. They apply to rules the basic principles of Linked Open Data and Linked Open Vocabularies: rules are designated by dereferencable URIs, expressed in W3C-compliant standards, and they can be linked to each other. Different formalisms are available to represent logical rules, such as SWRL[4] and SPIN[5]. SHACL[6] and its extension[7] are the latest W3C standard for rules representation. SHACL aims to represent constraints on an RDF graph, called "shapes", as well as deduction rules. SHACL rules, similarly to SPIN, can be based on SPARQL: it is possible to express a production rule in SHACL as a SPARQL CONSTRUCT query. Rules expressed in the SHACL formalism are used in the EDR approach we propose to provide an interoperable rule representation.

Logical rules being explicit deduction representations, they have been considered in IoT networks to express and share the correlation between sensor

[4] https://www.w3.org/Submission/SWRL/.
[5] https://www.w3.org/Submission/spin-modeling/.
[6] https://www.w3.org/TR/shacl/.
[7] https://www.w3.org/TR/shacl-af/.

observations and high-level symptoms since early work on the SWoT [20]. [19] lists numerous works using rules for context-awareness in the IoT. Inspired from the Linked Rules, the Sensor-based Linked Open Rules (S-LOR)[4] is dedicated to rules re-usability for deductions based on sensor observations. Deduction rules are a mechanism similar to Complex Event Processing (CEP) approaches such as [10], but the rule representation shifts from an ad-hoc rule format in CEP to a unified format in the SWoT.

In most existing approaches [4,26], rules are handled by Cloud nodes. An example of Industrial IoT (IIoT) use case enabled by Cloud-based semantic rules processing is presented in [25]. This paper proposes a self-configuring smart factory in which conveyors and machines produce data which is processed in a Cloud where user rules are used to make reconfiguration decisions. Rules are expressed in SWRL. Such architecture raises multiple issues, such as the cost of semantic reasoning that increases rapidly with the size of the Knowledge Base (KB) [11], and the impact on resiliency, since the Cloud node constitutes a single point of failure. A way of overcoming these issues is to consider Fog computing, defined by the open Fog consortium[8] as a "system-level horizontal architecture that distributes resources and services [...] anywhere along the continuum from Cloud to Things". The applicability of such a paradigm to the IoT, compared to pure Cloud computing, is particularly studied, e.g., by [13]. This work identifies key IoT requirements tackled by the Fog computing paradigm, namely **low latency**, **network topology dynamism**, and **scalability**. The constrained nature of Fog nodes (compared to Cloud nodes) must be taken into account: processing power or bandwidth are critical resources.

Most approaches for processing on constrained nodes focus on optimizations enabling such processing for a single node without considering the other, or in distributed cases processing placement is not dynamic: all nodes execute the same rules, or each a predefined static rule set. [3] shows how gateways are Fog nodes capable of enriching data: observations are initially produced by legacy devices in ad-hoc formats. It is the gateway, communicating with devices using protocols adapted to constrained environments, such as CoAP, that enriches the data before forwarding it towards the Cloud. Therefore, observations are enriched on the edge of the network, and only the Fog nodes in direct contact with legacy devices have to perform data enrichment. [5] proposes to execute a different type of rules in the Fog: Event Condition Action (ECA) rules associate a deduction with an action, which is used to automate the response of the system to a stimulus. However, the authors only consider one gateway executing the rules, and the ad-hoc rule format is not suited for rule exchange. Regarding processing distribution in existing work, the dynamic nature of IoT networks should be considered. The topology of a network evolves as devices connect, disconnect, or move geographically. Therefore, a viable distribution of rules at a given moment is not guaranteed to remain optimal in the future, and **the distribution strategy should be adapted to the evolution of the network topology**. [11] does not detail the mobility strategy used for its mobile nodes,

[8] http://openfogconsortium.org/.

and each node applies all the rules regardless of their relevance to the messages it aggregates. In [22], rule placement is static, in either Cloud or Fog nodes. [24] focuses on resource placement in a Fog-enabled IoT. The authors compute optimal deployment of application modules based on the representation of available resources on the Fog compared to requirements expressed by applications. Module positions are static, and computed at the time of deployment. Rules are deployed on gateways in an IIoT context in [7]. The rules themselves are not expressed using SW formalisms, and they are not dynamically assigned to Fog devices. However, ad-hoc mechanisms enable rule update at runtime. Rules are combined to a semantic engine proposed in [6] so that rules are expressed based on enriched data.

EDR differs from previous proposals in its focus on the dynamic rule deployment in the SWoT system at runtime, involving all the system's nodes. EDR is only based on either common knowledge, or **knowledge local to the node making the decision** to delegate processing to another node.

4 Enabling Rule Deployment Based on Their Representation

EDR is based on rule deployment in the Fog, in order to enable their placement on nodes according to a deployment optimization strategy, such as the proximity to the sensors producing the data they consume. The EDR core deployment technique is agnostic to the chosen strategy, and includes rule and data propagation. Rules are deployed **neighbor-to-neighbor** between Fog nodes, assuming the nodes are organized in a hierarchical topology as described in Sect. 4.1. The deployment of rules and the propagation of data are based on node local knowledge described with a vocabulary on which node functionalities rely. Both functionalities and the vocabulary are presented in Sect. 4.2. This vocabulary is embedded in rules as described in Sect. 4.3, where the modularity of rules is examined.

4.1 Assumptions, Underlying Architecture and Approach Overview

EDR is based on the hypothesis of a **hierarchical network topology**: nodes are organized in a tree-like structure, and only communicate with neighboring nodes. This assumption is made because such topologies are frequent in IoT networks, represented in studies such as [27], [1] (based on the oneM2M standard[9]), [23], or [22]. The tree root is the Cloud, leaves are devices, and nodes in between are Fog nodes. Applications are not part of the Cloud integrated to the IoT topology: they are executed remotely on personal devices such as smartphones or laptops. **Rules represent applicative needs**: when deductions from sensor observations are required by an application, it injects the rule in the network in order to be provided directly with the deductions, instead of being forwarded

[9] http://onem2m.org/.

raw data by the network and applying the rules itself. It is assumed therefore that Fog nodes can communicate with applications directly. Rules are initially submitted by applications to the Cloud node, so it is the only node they know a priori. The Cloud provides a unique permanent interface to the network, and the dynamic Fog topology underneath is therefore transparent for applications. Finally, we assume that since IoT data is strongly bound to a spacio-temporal context [14], the distribution of Fog nodes reflects the distribution of features observed by sensors. EDR is suitable for rules exploiting this spacio-temporal locality by correlating data sharing an identical context, for instance the activity of a machine and the level of particles in the same room needed in rule R4 of Table 1.

To ensure decentralization, the EDR approach is executed in parallel on each node able to perform reasoning in the topology, i.e. Cloud nodes and semantic-computing-enabled Fog nodes. EDR is a neighbor-to-neighbor approach: each node only communicates with its direct neighbors within the IoT network, i.e. its parent and children in the hierarchical topology. A parent node propagates a rule to its child if the parent considers that the child is empowered to apply the rule, based on the associated strategy. In the same way, a node n sends back a rule r to its parent if n considers it can no longer apply r. To enable the deployment of rules, nodes exchange messages describing their capabilities, e.g., their location or the type of data they observe. When a node makes a new deduction based on a rule, it sends the result to all the nodes interested, including the application that submitted the rule. A functional representation of an EDR node is provided in Fig. 2: each node has a local KB, where the information it has about its neighbors in the network topology is stored as well as the observations of the environment. The KB also contains the rules that have been sent to the node by either applications or other nodes, and these rules are used by the inference engine to update the KB. How the node functionalities are related to the KB in the core EDR mechanism to enable the propagation of observations and rules is first described in Sect. 4.2. The modular rule representation embedding the deployment strategy, and the updates of the KB they trigger, are detailed in Sect. 4.3.

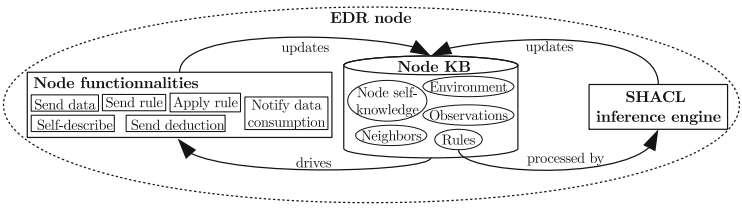

Fig. 2. EDR node functional overview

4.2 A Core Deployment Mechanism Based on a Dedicated Vocabulary

The hierarchical nature of the topology is captured in the EDR vocabulary[10]. The relation between a node n_p and its child n_c is expressed with the triplet <n_p,*lmu:hasDownstreamNode*,n_c>[11], based on a nomenclature presented in earlier work [18]. The inverse relation exists, to express the connection between a node n_c and its parent n_p: <n_c,*lmu:hasUpstreamNode*,n_p>.

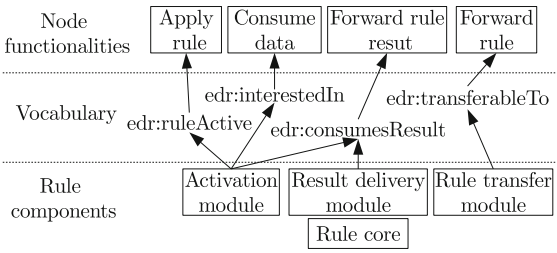

Fig. 3. Relation between node functions and rules modules

Node behavior is made quite simple on purpose, to decorrelate the rule application-specific deployment strategy, e.g., response time reduction or privacy enforcement, from the core deployment technique on which EDR is based, which is application-agnostic. The node functionalities are represented in Fig. 2, and their relation with the EDR vocabulary and the rule modules is shown in Fig. 3. Each functionality relies on dedicated triplets, and the node implements its behavior based on the description held in its KB. The first functionality of a node is to **apply rules**. When a node n receives a new observation, either from its own sensors or lower nodes, n executes the rules r stored in its KB if the description of rule r contains <r,*edr:isRuleActive*,true>. If processing the new observation with rule r by node n leads to a deduction δ, δ is sent to the nodes n' when <n',*edr:consumesResult*,r> in the KB of n: this is the **rule result delivery** functionality. Especially, the application that submitted the rule to the network, known as the rule originator, is a consumer of rule results, in order to enable deduction delivery to applications. Moreover, if the upper node of node n (denoted u_n) has declared its interest for the type of the new observation α_t, the observation if forwarded toward u_n. Observations are exchanged lazily: if a node n receives an observation of type α_t, and knows no other node interest in such type, the observation is not forwarded. Such interest is represented in node n KB with the triplet <u_n,*edr:isInterestedIn*,α_t>. Similarly, node n has to explicitly notify its children that it consumes the data type α_t to receive observations of

[10] Used namespaces: **edr**:<https://w3id.org/laas-iot/edr>, **lmu**:<https://w3id.org/laas-iot/lmu>, **sh**:<http://www.w3.org/ns/shacl>, **ex**:<http://example.org/ns>.

[11] Individuals such as $n-p$ and n_c are identified with a URI in the triplets.

this type. This active advertisement is denoted as the **data consumption** functionality: when node n has a triple in its KB such that $<n,edr{:}interestedIn,\alpha_t>$, it forwards this interest to its children. Finally, when a rule r is present in the KB of node n in a triplet $<r,edr{:}transferrableTo,n'>$, r if forwarded to node n' by n. This is the **rule transfer functionality**.

These core node functionalities are agnostic of the rule deployment strategy: they are solely based on the representation of themselves and their neighbors by nodes. How the triplets triggering node behaviors are injected in the knowledge base, and their relationship with the rules is detailed in Sect. 4.3.

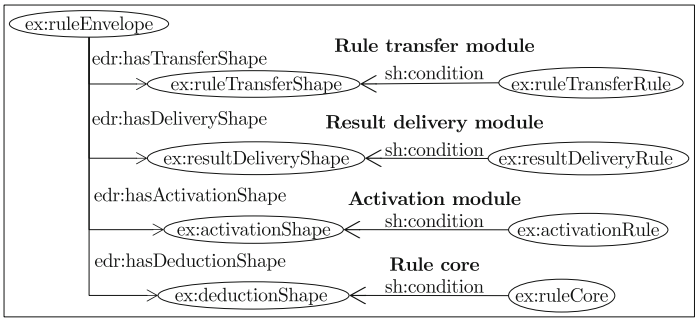

Fig. 4. Rule modules

4.3 Rule Representation and Deployment

EDR rules are composed of several modules, as represented on Fig. 3. Each of these modules enables some node functionalities: therefore, the intelligence is located in the rules, and not in the nodes. The deployment technique can therefore be parametrized at a fine granularity, for each rule (Fig. 4).

Rules are represented in SHACL, and the modules are based on SHACL advanced features named "SHACL rules". Each module is composed of two parts: a SHACL rule, that inserts deductions into the KB, and a SHACL shape that determines whether the rule is applied or not. An example rule is provided online[12]. In the remainder of this section, EDR is instantiated with a rule deployment and data propagation strategy enabled by rule modules referred to as EDR$_{\mathcal{PT}}$. EDR$_{\mathcal{PT}}$ extends the work initially proposed in [17], that does not consider the genericity of EDR. Therefore, in this previous work, no distinction is made between the generic EDR approach and its refinement with EDR$_{\mathcal{PT}}$. EDR$_{\mathcal{PT}}$ is based on the **property types** of data produced by nodes in order to reduce deduction delay for applications. EDR$_{\mathcal{PT}}$, dedicated to a specific strategy, is an instantiation of EDR. These properties can be either environmental properties captured by sensor observations (e.g., luminosity) or higher level properties deduced by other rules (e.g., safety).

[12] https://w3id.org/laas-iot/edr/iiot/r1.ttl.

The self-representation of nodes (stored in their KB) includes, but is not limited to, the types of property it observes. A node directly produces observations on a property if one of its children is a sensor dedicated to this property. Moreover, indirect observation of a property is enabled by a proxying mechanism. EDR$_{\mathcal{PT}}$ relies on a proxying mechanism such as in [12], where reasoning nodes act as proxy for the capabilities of legacy nodes unable to process enriched data. In EDR$_{\mathcal{PT}}$, each reasoning-enabled node n has a similar role, with respect to both its sensors and lower nodes that are proxied toward its parent. This mechanism makes a node aware of the types of properties produced by any node below its lower nodes while communicating only with its lower nodes, therefore ensuring the locality of its decisions. The production of observations by node n for a property type ρ_t is denoted <n,*edr:producesDataOn*,ρ_t>. When a child node connects to its parent, it includes such triplets in its self-description. If the parent node n was not a producer of the property type ρ_t, it includes a new triplet in its KB <n,*edr:producesDataOn*,ρ_t>, and forwards this triplet to its own parent. If node n was already a producer for rho_t, its capabilities remain unchanged, and the information propagation stops. A similar proxying mechanism is used to propagate interests: when a node n receives a message from its parent n' containing a triple <n',*edr:isInterestedIn*,ρ_t>, n sends a message to its children producing rho_t (if any) containing a triple <n,*edr:isInterestedIn*,ρ_t>. The knowledge of nodes about their environment is therefore limited to their neighborhood, enabling purely local decisions.

The knowledge of property types observed by remote nodes is used for rule deployment in EDR$_{\mathcal{PT}}$. The operational part of the rule, containing the application-dedicated inference, is referred to as the **rule core** module. It is in this part of the rule that the properties observed by the node are exploited to make deductions. The rule core has a structure similar to other rule modules, with a conditional shape and a deduction SHACL rule. The condition of the rule core only tests that the triple <r,*edr:isRuleActive*,true> is in the node's KB. To materialize the property types featured in rule core in the following, we use the notations $body\_t(r_x) = \{\gamma_1, ..., \gamma_{n'}\}$ and $head\_t(r_x) = \{\delta_1, ..., \delta_{m'}\}$ where γ_i designates the property type of Γ_i, and δ_j the property type of the deduction Δ_j. It should be noted that not all Γ_i or Δ_j used in the rule are relevant to the EDR$_{\mathcal{PT}}$ approach. For R1, an illustrative rule introduced in the use case Sect. 2, $body\_t(R1) = \{luminosity, presence,$ *machine state*$\}$, and $head\_t(R1) = \{$*low machine visibility*$\}$. When a node receives a rule, or when the knowledge it has on its neighbors is updated, it triggers a reasoning step to process rule modules.

The **rule transfer module** determines on which remote nodes the rule may be deployed. A child node n' is considered for the forwarding of a rule r from node n if, $\forall \rho_t \in body_t(r)$, <$n'$,*edr:producesDataOn*,$rho_t$>. This condition is expressed as a SPARQL query as part of the SHACL rule being the conditional part of the rule transfer module. The deduction part of the module infers <r,*edr:transferrableTo*,n'>, enabling the rule forwarding mechanism of the node.

The **activation module** detects if the current node is suitable to apply the rule itself. To apply a rule r, a node n must be the lowest common ances-

tor to the producers of property types in the rule body. Such node has a set $\mathcal{P}$ of children partially producing the rule head. Individually, none of the children produce all the elements of the rule head, but combined, their productions enable the processing of the rule. It is characterized as such: $\exists\mathcal{P}$, such as $\forall n' \in \mathcal{P}$, $<n,lmu{:}hasDownstreamNode,n'>$ and $\exists\{\rho_t, \rho'_t\} \subseteq body(r)$, $<n',edr{:}producesDataOn,\rho_t>$ and $\neg\exists\ <n',edr{:}producesDataOn,\rho'_t>$, and $\forall\rho_t \in body(r), \exists n' \in \mathcal{P}$, $<n',edr{:}producesDataOn,\rho_t>$. If the conditional part of rule r activation module determines that the current node is suitable to apply r, some deductions are inferred. The activity of rule r is made explicit by the triplet $<r,edr{:}isRuleActive,$true$>$, and the nodes $n' \in \mathcal{P}$ are identified by $<n',edr{:}partialDataProvider,\rho_t>$. The interest of the rule originator o is also denoted with $<o,edr{:}consumesResult,r>$. These inferences enable both the **rule application** and the **rule result delivery mechanisms** as described in Sect. 4.2.

The **result delivery module** enables the forwarding of deductions to other nodes that are not the originator of the rule, such as the parent n' of a node n if n' applies a rule r' that consumes the deductions made by a rule r applied by n. In EDR$_{\mathcal{PT}}$, the condition of the result transfer module checks if a node expressed interest for the type of deductions yielded by the rule. If there exists a triple $<n',edr{:}interestedIn,\rho_t>$, with n' a remote node and ρ_t an element of the rule r's head $head(r)$, then the result transfer module infers that $<n',edr{:}consumesResult,r>$.

It is worth noting that rule modules only need to be evaluated when the rule is received, or when the topology evolves, e.g., with new productions by children or new consumptions by parents. On the other hand, the rule core must be computed each time a new observation is received by the node. To reduce the computation load, and to only process rule modules when needed, a SHACL functionality is used: the reasoner does not consider shapes or rules r such that $<r,sh{:}deactivated,$true$>$. The modules of a rule r are therefore only activated for a reasoning step when r is received, or when the topology evolves.

5 Experiments

To evaluate EDR$_{\mathcal{PT}}$, the smart factory use case described in Sect. 2 is simulated on a setup detailed in Sect. 5.2. The purpose of the EDR (instantiated by EDR$_{\mathcal{PT}}$) approach we propose is to directly feed rule deductions to applications. However, for comparison purposes, multiple deduction delivery mechanisms are proposed and compared in Sect. 5.1.

5.1 Deduction Delivery Mechanisms

Unlike rule deployment strategies, deductions delivery mechanisms are decorrelated from the rules: they are variations of the "Forward rule result" functionality described in Sect. 4.2. Therefore, the propagation of rules, the deductions they yielded and data is described as intended according to ad-hoc strategies (here,

EDR$_{\mathcal{PT}}$) through the EDR vocabulary. However, for experimental purpose, this propagation can be altered at the node level, preventing rule deployment or rerouting deduction delivery. Five deduction delivery mechanisms are compared in our experiments:

- **Cloud-Indirect-Raw (CIR)** is the baseline approach: the rules are only kept in the Cloud, and raw observations are forwarded neighbor-to-neighbor from the nodes that collect them toward the central node. The Cloud then delivers deductions to applications. Applications are notified by the Cloud, and not by Fog nodes, in all the following strategies except the last one.
- **Cloud-Direct-Raw (CDR)** is also an approach where rules are not deployed, and only processed in the central Cloud. However, the interest proxying mechanism presented in Sect. 4.3 is altered: node that are not the upper node in the hierarchy propagate the interests they receive without proxying them. Therefore, the observation producers directly send raw observations to the Cloud node, where they are used for rule-based deductions.
- **Cloud-Direct-Processed (CDP)** is a hybrid approach, where rules are deployed in the Fog to be processed as far as possible in the hierarchy, but where deduction are delivered directly to the Cloud instead of sending them to applications. To do so, when forwarding a rule it has received, the Cloud node declares itself as the originator instead of the application. Deductions can also be propagated in the Fog if a node explicitly expressed its interest. Processing rules in the Fog means that the propagation of observations is limited to the Fog nodes applying rules consuming such observations.
- **Cloud-Indirect-Processed (CIP)** is another hybrid strategy for EDR$_{\mathcal{PT}}$: rules are processed in the Fog, and the results are propagated neighbor-to-neighbor towards the Cloud before being delivered to applications. To modify the result delivery behavior, whenever a node propagates a rule, it declares itself as the originator of the said rule.
- **Application-Direct-Processed (ADP)** is the purely decentralized strategy that we propose for EDR$_{\mathcal{PT}}$, where rules are processed in the Fog and deductions are delivered directly to applications that submitted the rules. In this case only, a deduction that has been inferred in the network will not be hosted by the Cloud before being delivered.

5.2 Experimental Setup and Implementation

Variations of a reference building's device network topology, described in the use case in Sect. 2 and displayed in Fig. 1, are simulated. The root node at depth 0 is the Cloud server while other nodes are Fog nodes. Sensors, not represented on the figure, are arbitrarily connected under nodes. Each sensor pushes a random observation to its parent every two seconds. To assess the distributed nature of EDR$_{\mathcal{PT}}$, and its suitability for constrained Fog nodes, the experimental setup includes a Raspberry Pi 3, a laptop and a server, described in Table 2. Each physical machine hosts multiple virtual nodes, composed of an HTTP server, a KB, a SHACL-enabled SPARQL engine, and a code base[13].

[13] The code is available at https://framagit.org/nseydoux/edr.

To measure the responsiveness of applications enabled by EDR, the **response time between the reception** of the piece of data **and the deductions** they triggered is measured. Precisely, the response time for the processing of a rule is characterized as the time difference between the moment when the most recent data used in the body of the rule is produced, and the moment when the rule head is received by the application. A dedicated timestamp is associated to each observation once it has been enriched, to avoid any impact of the enrichment process on the measure. For instance, if a conveyor speed observation measured at t_1 and a machine speed observation measured at t_2 trigger a deduction by R7 received by the originating application at t_3, the response time for this particular deduction will be $t_3 - max(t_1, t_2)$.

Table 2. Experimental setup

	RAM	Cores	CPU
Server	32 GB	32	3.0 GHz
Laptop	16 GB	8	2.6 GHz
RPi 3	1 GB	4	1.4 GHz

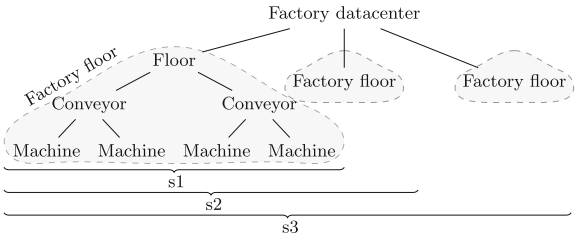

Fig. 5. Simulation topology s*

Experimental measures showed that, for each simulation, the number of deductions is consistant between centralized and distributed approaches: there is no knowledge loss when applying EDR$_{\mathcal{PT}}$ under our assumptions of bound between the Fog topology and the correlation between data.

5.3 Scalability of EDR$_{\mathcal{PT}}$

To assess the scalability of the proposed strategy for EDR, performances have been measured on three topologies, denoted s1, s2 and s3[14], and collectively as s*, as represented on Fig. 5. All s* topologies mimic the use case architecture presented in Fig. 1, with variations in the number of floors. A floor is constituted of two conveyors, each of which supports two machines, with sensors distributed as shown on a JSON blueprint provided online[15], leading to a total of 30 nodes (including both reasoning nodes and sensors). The rules described in Sect. 2 are used. The number of nodes is increased by duplicating floors: s1 has one, s2 two, and s3 three floors, for a total number of respectively 31, 61 and 91 nodes. After presenting deduction delay measures, a deeper analysis of the causes for these delays is provided.

[14] Topology representations are available at https://w3id.org/laas-iot/edr/iiot/scala_syndream/clone_f_⟨0,1,2⟩.ttl respectively.

[15] https://w3id.org/laas-iot/edr/iiot/clone_f_0_blueprint.json.

Due to scaling issues, results are separated in two figures: on Fig. 6, results are shown on the left for deduction propagation strategies CIR and CDR, where observations are sent to the Cloud where they are processed according to rules, and on the right for deduction propagation strategies CIP, CDP and ADP, where $\text{EDR}_{\mathcal{PT}}$ is implemented and rules are distributed in the network.

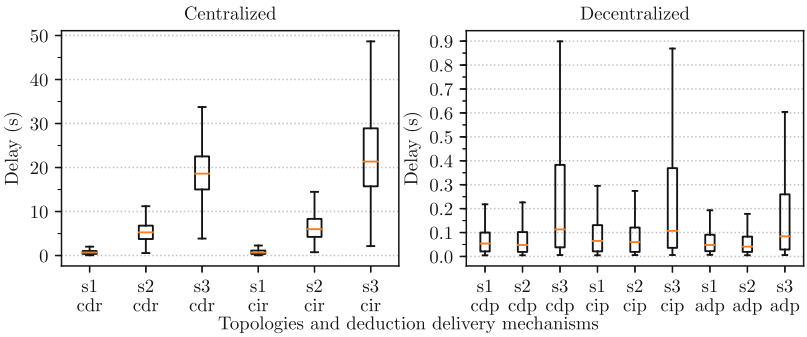

Fig. 6. Scalability measures

The gain in scalability provided by the decentralized approaches appears in the results. In topology s1, the discrepancy between response time for distributed and centralized approaches is reduced, with a median around 0.65 s for CIR and CDR, and 0.065 s for CDP, CIP and ADP. However, in topologies s2 and s3, the gap between centralized and distributed approaches increases dramatically. An increase is also observed for distributed deduction strategies, but it is smaller. Approaches promoting direct communication, i.e. CDR and CDP, perform better that their indirect counterparts, respectively CIR, CIP. This is an expected result, as direct communication reduces the number of hops required for a message (be it an observation or a deduction) to reach its target.

To analyze closely the cause for the increased delay, the journey of a message has been broken down in discrete timestamped events. The first event related to a message is its construction, either by enrichment of an observation or by achieving a deduction. In order to be propagated in the network, a message might be sent from a node n to another node n', which is identified as two events: the sending from node n, and the reception by node n'. Multiple hops are registered, from the first node responsible for the message creation toward any node that is interested in the message content for deduction. When a message is received by a node n, n starts a reasoning step where it tries to make new deductions based on the rules in its knowledge base. Events are logged at the beginning and at the end of reasoning. To detail the delay for each deduction, the journey of the most recent observation leading to the deduction is reconstructed. This journey is built by identifying all consecutive events related to the piece of data leading to the deduction, from its initial enrichment to its processing leading to the deduction, and the delivery of said deduction to the application. The average

composition of delays is broken down in Fig. 7, after normalization: each bar represents the share of the delay type in the global delay. Three types of delay have been characterized:

- **Transfer delays**, measured between the emission and the reception of a message. This delay is both impacted by the quality of the network link between two nodes, but also by the processing speed of the recipient: the transfer is considered completed when the recipient declares the reception at the software level, and it is not measured at the network layer. When the message is transferred through multiple hops, the delays are summed.
- **Reasoning delays**, measured between the beginning and the end of a reasoning step. To prevent multiple deductions with the same message in distributed cases, the reasoning step is applied before propagation, and the observation is marked with the rules it has been processed by. Reasoning delays are summed if the same message is processed with different rules across the topology.
- **Idle delays**, measured between the reception of a message and its processing, or between the reasoning step and the propagation of deductions.

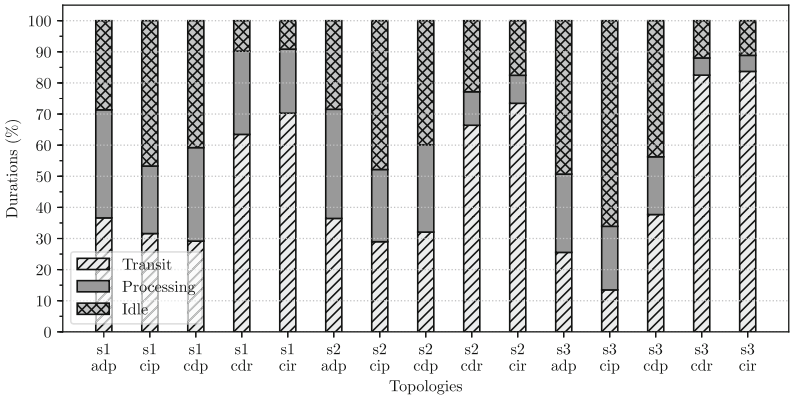

Fig. 7. Breakout of delays (normalized)

A trend that can be observed in the breakout is the increase of the share of transfer time in centralized strategies compared to decentralized ones. An explanation for this phenomenon is the saturation of the network link, combined to an overhead on the central node induced by the necessity to perform all the reasoning. The central node has less CPU time available to declare reception of messages, and therefore the time between the emission event and the reception event is increased. Overall, the limited increase of delays and the balance of the delays breakdown in the distributed settings support our claim that $\text{EDR}_{\mathcal{PT}}$ is a scalable approach to rule-base reasoning in the Fog.

6 Conclusion and Future Work

In this paper, we presented EDR, a distributed rule deployment technique for cooperative semantic computing associating Cloud and Fog resources for smart heterogeneous IoT networks. The elaborated solution leverages the Cloud and Fog complementarity to implement an efficient deployment and propagation approach for data, rules and deductions respectively. Such a solution can facilitate the development of SWoT systems and the associated semantically-enabled IoT applications with non-functional requirements related to performance or privacy. The approach we propose is more general than any specific rule deployment strategy chosen by applications according to their requirements: in our contribution, rule deployment strategies are embedded in the rules themselves. We instantiate EDR by EDR$_{\mathcal{PT}}$with a deployment strategy dedicated to scalability and response time. EDR$_{\mathcal{PT}}$ allows rules to be executed on Fog nodes close to the sensors producing the type of data these rules consume.

The scalability of EDR$_{\mathcal{PT}}$ is studied based on a smart factory use case, where seven rules are propagated across three instances of a reference hierarchical tree-like topology according to five different deduction delivery mechanisms. The results, when measuring responsiveness from an application standpoint, support the performance superiority of the partially or totally distributed deduction delivery mechanism, where rule computation is balanced between Cloud and Fog nodes, compared to the centralized ones, where the whole computation is achieved on Cloud resources.

In future work, we will consider situations where direct communication between Fog nodes and the Cloud or applications is not possible. It is especially interesting in deployments where nodes communicate over ad-hoc networks where border gateways connecting networks can be used as relays to forward observations and deductions. In such a topology, the Application-Direct-Processed and Cloud-Direct-Processed delivery mechanisms cannot be applied, and proposing a new approach derived from Cloud-Indirect-Processed may allow the relaxation of our hypothesis requiring direct communication between the Fog and application. The deployment of rules in the Fog can also be driven by privacy requirements, a predominant concern in the IoT community[16]. Shifting the paradigm from the concentration of data in remote centralized nodes to the propagation of processing close to data producers and consumers enforces the locality of data processing. We intend to elaborate a privacy-aware deployment strategy, to measure how the overhead of traffic, compared to the solution proposed in the present paper, impacts performances, and how to find a trade-off between privacy and performances.

[16] https://www.slideshare.net/kartben/iot-developer-survey-2018.

References

1. Alaya, M.B., Medjiah, S., Monteil, T., Drira, K.: Toward semantic interoperability in oneM2M architecture. IEEE Commun. Mag. **53**(12), 35–41 (2015). https://doi.org/10.1109/MCOM.2015.7355582
2. Ali, M.I., et al.: Real-time data analytics and event detection for IoT-enabled communication systems. Web Semant.: Sci. Serv. Agents World Wide Web **42**, 19–37 (2017). https://doi.org/10.1016/j.websem.2016.07.001
3. Desai, P., Sheth, A., Anantharam, P.: Semantic Gateway as a Service architecture for IoT Interoperability. Netw. Internet Arch., 16 (2014). https://doi.org/10.1109/MobServ.2015.51
4. Gyrard, A., Serrano, M., Jares, J.B., Datta, S.K., Ali, M.I.: Sensor-based Linked Open Rules (S-LOR): An Automated Rule Discovery Approach for IoT Applications and its use in Smart Cities. In: International Conference on World Wide Web Companion, pp. 1153–1159 (2017). https://doi.org/10.1145/3041021.3054716
5. Kaed, C.E., Khan, I., Berg, A.V.D., Hossayni, H., Saint-Marcel, C.: SRE: semantic rules engine for the industrial Internet-of-Things gateways. IEEE Trans. Ind. Inform. **14**(2), 715–724 (2018). https://doi.org/10.1109/TII.2017.2769001
6. Kaed, C.E., Khan, I., Hossayni, H., Nappey, P.: SQenloT: semantic query engine for industrial Internet-of-Things gateways. In: 2016 IEEE 3rd World Forum on Internet of Things, WF-IoT 2016, pp. 204–209 (2016). https://doi.org/10.1109/WF-IoT.2016.7845468
7. 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 Trans. Ind. Inform. **14**(2), 715–724 (2018). https://doi.org/10.1109/TII.2017.2769001
8. Khandelwal, A., Jacobi, I., Kagal, L.: Linked rules: principles for rule reuse on the web. In: Rudolph, S., Gutierrez, C. (eds.) RR 2011. LNCS, vol. 6902, pp. 108–123. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-23580-1_9
9. Kiljander, J., et al.: Semantic interoperability architecture for pervasive computing and Internet of Things. IEEE Access **2**, 856–873 (2014). https://doi.org/10.1109/ACCESS.2014.2347992
10. Li, Z., Chu, C.H., Yao, W., Behr, R.A.: Ontology-driven event detection and indexing in smart spaces. In: IEEE International Conference on Semantic Computing, pp. 285–292 (2010). https://doi.org/10.1109/ICSC.2010.63
11. Maarala, A.I., Su, X., Riekki, J.: Semantic reasoning for context-aware internet of things applications. IEEE Internet Things J. **4**(2), 461–473 (2017)
12. Nikoli, S., Penca, V., Konjovi, Z.: Semantic web based architecture for managing hardware heterogeneity in wireless sensor network. Int. J. Comput. Sci. Appl. **8**(2), 38–58 (2011)
13. Patel, P., Intizar Ali, M., Sheth, A.: On using the intelligent edge for IoT analytics. IEEE Intell. Syst. **32**(5), 64–69 (2017). https://doi.org/10.1109/MIS.2017.3711653
14. Perera, C., Zaslavsky, A., Christen, P., Georgakopoulos, D.: Context aware computing for the internet of things: a survey. IEEE Commun. Surv. Tutor. **16**(1), 414–454 (2014). https://doi.org/10.1109/SURV.2013.042313.00197
15. Pfisterer, D., et al.: SPITFIRE: toward a semantic web of things. IEEE Commun. Mag. **49**(11), 40–48 (2011). https://doi.org/10.1109/MCOM.2011.6069708
16. Poslad, S., Middleton, S.E., Chaves, F., Tao, R., Necmioglu, O., Bugel, U.: A semantic IoT early warning system for natural environment crisis management. IEEE Trans. Emerg. Top. Comput. **3**(2), 246–257 (2015). https://doi.org/10.1109/TETC.2015.2432742

17. Seydoux, N., Drira, K., Hernandez, N., Monteil, T.: Reasoning on the edge or in the cloud? Internet Technol. Lett., e51
18. Seydoux, N., Drira, K., Hernandez, N., Monteil, T.: Capturing the contributions of the semantic web to the IoT: a unifying vision. In: Maleshkova, M., Verborgh, R., Gyrard, A. (eds.) Proceedings of the Second SWIT Workshop Co-located with ISWC. CEUR Workshop Proceedings, vol. 1930 (2017)
19. Sezer, O.B., Dogdu, E., Ozbayoglu, A.M.: Context-aware computing, learning, and big data in internet of things: a survey. IEEE Internet Things J. **5**, 1–27 (2018)
20. Sheth, A., Henson, C., Sahoo, S.S.: Semantic sensor web. IEEE Internet Comput. **12**(4), 78–83 (2008). https://doi.org/10.1109/MIC.2008.87
21. Shi, W., Dustdar, S.: The promise of edge computing. Computer **49**(5), 78–81 (2016)
22. Su, X., et al.: Distribution of semantic reasoning on the edge of Internet of Things. In: IEEE UbiComp, p. 79, November 2018
23. Szilagyi, I., Wira, P.: Ontologies and semantic web for the Internet of Things - a survey. In: IECON. IEEE (2016)
24. Taneja, M., Davy, A.: Resource aware placement of IoT application modules in Fog-Cloud Computing Paradigm. In: 2017 IFIP/IEEE Symposium on Integrated Network and Service Management, pp. 1222–1228. IEEE, May 2017. https://doi.org/10.23919/INM.2017.7987464
25. Wang, S., Wan, J., Li, D., Liu, C.: Knowledge reasoning with semantic data for real-time data processing in smart factory. Sensors (Switzerland) **18**(2), 1–10 (2018). https://doi.org/10.3390/s18020471
26. Xu, G., Cao, Y., Ren, Y., Li, X., Feng, Z.: Network security situation awareness based on semantic ontology and user-defined rules for internet of things. IEEE Access **5**, 21046–21056 (2017)
27. Zanella, A., Bui, N., Castellani, A., Vangelista, L., Zorzi, M.: Internet of Things for smart cities. IEEE Internet Things J. **1**(1), 22–32 (2014). https://doi.org/10.1109/JIOT.2014.2306328