A Distributed Scalable Approach for Rule Processing: Computing in the Fog for the SWoT

Nicolas Seydoux*†, Khalil Drira†, Nathalie Hernandez* and Thierry Monteil†
*IRIT,
Maison de la Recherche, Univ. Toulouse Jean Jaurès,
5 allées Antonio Machado, F-31000 Toulouse
email: {name.surname}@irit.fr
†LAAS-CNRS,
Université de Toulouse, CNRS, INSA, Toulouse, France
email: {name.surname}@laas.fr

***Abstract*—The development of the Semantic Web of Things (SWoT) is challenged by the nature of IoT deployment architectures, where constrained devices collect data processed remotely by powerful Cloud servers. Such a deployment pattern introduces bottlenecks constituting a hurdle for scalability, and increases response time. This hinders the development of a number of critical and time-sensitive applications. Enabling the deployment of the Semantic Web stack closer to the constrained devices of the IoT may foster the development of time-sensitive interoperable applications, while reducing forwarding the user data to remote third party Cloud servers. The approach we develop in this paper is a contribution towards this direction, and aims to enable rule-based reasoning closer to sensors producing IoT data. For this purpose, we define a distributed scalable semantic processing algorithm by dynamically propagating deduction rules on Fog nodes. Our goal is to shorten the time needed to deliver high level information deduced from the collected data. This approach is evaluated on a smart building use case where both distribution and scalability have been considered.**

I. Introduction

The maturity of Internet of Things (IoT) communication technology is leading to a wide variety of industrial and societal applications involving responsiveness and privacy requirements, as for e-health or home automation. However, architecting IoT service platforms and applications faces new challenges emerging from data heterogeneity on the one hand and real-time decision management complexity on the other. Such challenges can be addressed by investigating semantic interoperability and automated reasoning techniques. Semantic interoperability allows IoT systems to share a common meaning over exchanged data. These requirements, crucial to IoT applications, are fostering the adoption of the Semantic Web principles and technologies as interoperability enablers [1]. A new domain has emerged from the interaction between the IoT and the Semantic Web, the Semantic Web of Things (SWoT) [2], raising additional challenges. The number of connected devices [3] 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, Semantic Web technologies are resources-consuming, and go beyond the capacities of constrained IoT devices, leading to centralized deployment approaches. The Semantic Web stack components are executed on the Cloud, where powerful servers collect and process IoT data before feeding applications with curated data, analytics results and decision instructions [4]. 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 [3] and degrades quality of service for IoT applications. It may hamper the development of a variety of applications and inhibit the deployment of time-critical applications. Our objective is to face these issues by adopting the Fog computing paradigm for the SWoT infrastructure by considering on devices located between constrained sensors and the Cloud as initiated in [5]. The aim is not to replace the Cloud by the Fog, but to use them as complementary computing resources providers. 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 [5], [6]. These limited computation capabilities are also leveraged by the development of dedicated semantic web libraries, such as μJena[1].

IoT applications, e.g 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 in order to provide high-level information relevant to the application end user [7]. Rules can be used to capture the specific needs of applications by representing and sharing dedicated deduction intent [8]. Existing IoT architectures mainly process rules on the Cloud [9]. In this paper we propose an approach in which, aware of application needs, Fog nodes can **produce information of concern for applications directly** from sensor observations, instead of being required to provide raw data to Cloud nodes for processing and delivery to applications. Rules applied on nodes closer to sensors can yield deductions faster compared to 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 [6] [10]. However, identifying which Fog nodes should process which rule in order to provide applications with the required results while

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

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 in order to **increase application responsiveness** in a **scalable** manner. We propose Emergent Distributed Reasoning (EDR), an approach to semantic processing based on dynamic rule propagation in a device network. EDR relies on local knowledge, where each node appropriately propagates reasoning rules toward the edge of the device network, closer to sensors, in order to accelerate their processing.

The remainder of this paper is organized as such: Section II describes existing approaches. Section III details the EDR approach, with the associated hypothesis and knowledge representations. EDR is evaluated in Section IV, using a smart building use case to drive the experiments. Finally, the paper is concluded in Section V.

II. Related work

As the concern of the proposed approach is to improve responsiveness when deducing application dedicated information from IoT data, state-of-the-art work dealing with logical rules for the Semantic Web, 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 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 [11]. 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[2] and SPIN[3]. SHACL[4] and its extension[5] are the latest W3C recommendations 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. **SHACL rules are used in the EDR approach we propose**.

Logical rules being explicit deduction representations, they have been considered in IoT networks to express and share the correlation between sensor observations and high-level symptoms since early work on the SWoT [12]. [8] lists numerous works using rules for context-awareness in the IoT. Inspired from the Linked Rules, the Sensor-based Linked Open Rules (S-LOR) [9] 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 [13], but the rule representation shifts from an ad-hoc rule format in CEP to a unified format in the SWoT.

In most existing approaches [9] [14], rules are handled by Cloud nodes. Such architecture raises multiple issues, such as the cost of semantic reasoning that increases rapidly with the size of the Knowledge Base (KB) [6], 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[6] 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 [5]. 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 at the individual scale, and in distributed cases processing is statically assigned. [15] 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. [16] 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 in this work, only one gateway executing the rules is considered, and the ad-hoc rule format is not suited for rule exchange. Regarding processing distribution in existing work, the dynamic nature of IoT deployments 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 deployment**. [6] does not detail the mobility strategy used for its mobile nodes, and each node applies all the rules regardless of their relevance to the messages it aggregates. In [10], rules are statically assigned to either Cloud or Fog nodes. [17] 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. EDR differs from previous proposals in its focus on the dynamic reconfiguration of rule deployment at runtime involving all the managed system's nodes.

The purpose of distributing data processing is to provide scalability and to prevent the appearance of a single point of failure. Processing should therefore be distributed based on local decisions made by each node, rather than based on a central controller (which would still be a single point of failure). The algorithm proposed in Section III is only based on either common knowledge, or **knowledge local to the node making the decision** to delegate processing to another node.

[2]https://www.w3.org/Submission/SWRL/

[3]https://www.w3.org/Submission/spin-modeling/

[4]https://www.w3.org/TR/shacl/

[5]https://www.w3.org/TR/shacl-af/

[6]http://openfogconsortium.org/

III. Distributing semantic processing by rule propagation in an IoT context

The core principle of EDR is to propagate semantic processing in an IoT network to enable decentralized rule-based reasoning. Rules are propagated between nodes in order to be processed as close as possible to sensors producing the data, so that deductions are directly provided to applications.

A. Assumptions and underlying architecture

EDR is based on the hypothesis of a **hierarchical network topology**: nodes are organised in a tree-like structure, and only communicate with neighboring nodes, i.e. their parent or children. This assumption is made because such topologies are frequent in IoT networks, represented in studies such as [18], [19] (based on the oneM2M standard[7]), [20], or [10]. 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 deployment: they are executed remotely on personal devices such as smartphones or laptops. **Rules represent applicative needs**: when an application requires deductions to be made from sensor observations, instead of being provided with raw data by the network and apply the rules themselves, they inject the rule in the network to be provided directly with the deductions. 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. It prevents them from encountering issues when communicating with Fog nodes, by nature dynamic and not guaranteed to be permanently available.

In order to ensure decentralization, the algorithm of the EDR approach is executed in parallel on each node able to perform reasoning in the topology, i.e. Cloud and semantic computing-enabled Fog nodes. It is based on a peer-to-peer approach, where each node only communicates with its direct neighbours, 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 able to apply the rule. In the same way, a node sends back a rule to its parent if it considers it can no longer apply it. This decision process is described in Section III-C. It relies on the local knowledge a node has about its neighbors. The content of nodes KBs is described in Section III-B, and the decision-making algorithm is detailed in Section III-D. The expressiveness of the supported rules is described in Section III-C, once the rules themselves have been introduced in Section III-B1.

B. Knowledge considered by nodes applying EDR

1) Rules representation: The rules considered by EDR are predicate logic rules used to deduce high-level information. Let r_x be such a rule noted as $r_x : \Gamma_1 \wedge ... \wedge \Gamma_n \rightarrow \Delta_1 \wedge ... \wedge \Delta_m$, where $\Gamma_1 \wedge ... \wedge \Gamma_n$, designated as the body of r_x, is a conjunction of conditions and $\Delta_1 \wedge ... \wedge \Delta_m$, designated as the head of r_x, is a conjunction of deductions. To represent and store rules in each node's KB, SHACL formalism is used.

Let us consider a home automation example application needing to be informed about the level of comfort of each room. The high level information useful for this application can be expressed in a rule, made simple for the sake of clarity, such as "**If** in a room the observed luminosity is above 400Lx and the observed temperature is between 20 and 25 degrees, **then** this room is comfortable". This rule $r_{comfort}$ can be represented with predicates as $room(x) \wedge luminosity(y) \wedge observation(x,y) > 400 \wedge temperature(z) \wedge 20 <= observation(z,x) <= 25 \Rightarrow confortable(x)$. This rule expressed in SHACL is available online[8].

Additionally to SHACL standard vocabulary, the EDR approach uses dedicated metadata in order to manage rules, noted with the prefix *edr:*[9] from now on. EDR is driven by the rules' predicates expressing types of **properties of the environment**. These properties can be either environmental properties captured by sensor observations (e.g. luminosity) or higher level properties deduced by other rules (e.g. comfort). Triples indicating the **types** of properties involved in the rule's body and head are added to the rule description. The intuition is that, based on this description, a node will be able to compare the type of properties needed to apply the rule to the property types it manages, without needing to process it.

To manipulate these types 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 approach. For our illustrative rule, $body\_t(r_{comfort}) = \{luminosity, temperature\}$, and $head\_t(r_{comfort}) = \{comfortable\}$. Property types in rules' head and body are represented respectively with the predicates *edr:hasRuleBody* and *edr:hasRuleHead* in the rule description.

The description also contains information about the application(s) interested in the rule. When an application app submits a rule r_x to the network, app is called the originator of r_x. It is attached to the rule metadata with the *edr:ruleOriginatedFrom* predicate. r's description also contains an HTTP endpoint denoted with the predicate *edr:originatingEndpoint*. With this information, a node applying the rule is able to deliver the deductions directly to the consuming application.

The rule's metadata introduced in this section are used by nodes in the propagation process at the core of EDR. They are compared as described in Section III-D to the knowledge nodes have on themselves and their neighbors, which are respectively described in Sections III-B2 and III-B3.

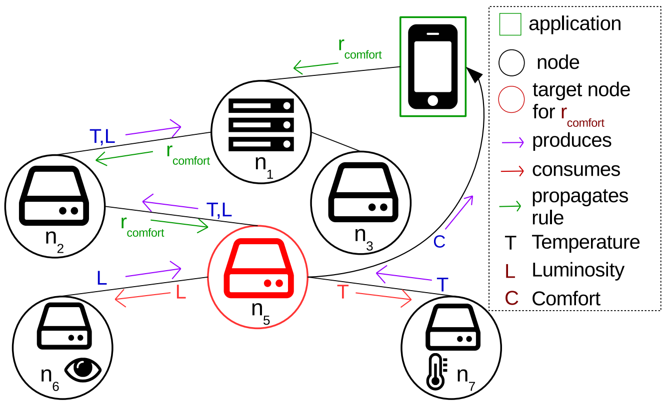
2) Node's knowledge on itself: A node n has in its KB information about the property types of the data it produces, denoted $own\_productions(n)$. Data produced by node n is either collected by sensors to which n is directly connected, or obtained as deductions when n applies a rule. When a reasoning-enabled node is connected to a sensor, it enriches the raw observation, and propagates the enriched observation on the network, which ensures that the observation is only enriched once. Fig. 1 is an example of topology in which EDR

[7] http://onem2m.org/

[8] https://w3id.org/laas-iot/rules/comfort/comfortableSpot.ttl

[9] https://w3id.org/laas-iot/edr

Fig. 1: Example of propagation for $r_{comfort}$



is used to execute the $r_{comfort}$ rule in the Fog, used in this section for illustrative purposes. In this topology, the Fog node n_6 is connected to a luminosity sensor and enriches its production therefore $own\_productions(n_6) = \{luminosity\}$. An example of enriched observation is available online[10]. Observations and devices are described in each node's KB using the IoT-O ontology[11] [21] for our experiments (Section IV), but the proposed algorithm does not depend on the ontology used to describe data, as long as the same ontology is used to express the rules and their metadata.

3) Node's knowledge on the topology: The transmission of rules among nodes organized by EDR is driven by the knowledge each node has on the network around itself. Node n knows its parent in the network tree-like hierarchy, noted $Upper(n)$, and its children, referred to as $Lower(n)$. On Fig. 1, $Lower(n_5) = \{n_6, n_7\}$, and $Upper(n_5) = \{n_2\}$.

In [22], reasoning nodes act as proxy for the capabilities of legacy nodes unable to process enriched data. In EDR, each reasoning-enabled node n has a similar role, with respect to both its sensors and $Lower(n)$ that are proxied toward $Upper(n)$. 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. Let us define $productions(n) = own\_productions(n) \cup productions(Lower(n))$. Node n announces itself to $Upper(n)$ as a producer of $\rho_i, \forall \rho_i \in productions(n)$ (ρ_i being the property type of data produced by a sensor or a lower node connected to n). For instance, on Fig. 1, $productions(n_5) = \{temperature, luminosity\}$

In order to limit unnecessary exchanges of data, data is exchanged lazily: node n has to explicitly advertise its interest for a property type ρ_j to $Lower(n)$ in order to be notified when new observations are received or new deductions are made. In Fig. 1, n_6 announced to n_5 that it produced $luminosity$, and n_5 notified n_6 of its interest in $luminosity$ to receive observations. The interest of node n in a property of type ρ_i is only propagated to node n' such that $n' \in Lower(n) \wedge \rho_i \in productions(n')$. The set of property types in which node n is interested is denoted $consumptions(n)$. A node is interested in a property type ρ_i when it is in charge of applying a rule whose body includes ρ_i. Identifying if $\rho_i \in body\_t(r)$ is based on URI comparisons. Furthermore, to make data propagation possible and based on local knowledge only, the proxying approach used for nodes productions is also used for node consumptions. Therefore, node n is also interested in ρ_i if its upper node notifies n that it consumes ρ_i.

[10] https://w3id.org/laas-iot/rules/observations/enriched_data.ttl
[11] https://www.irit.fr/recherches/MELODI/ontologies/IoT-O

C. Combining topology knowledge and rules metadata

The purpose of EDR is to **transfer each rule to the lowest possible node in the architecture**, to be applied as early as possible. The propagation of a rule r_x from node n to node n' is considered relevant in two cases. Either the rule r_x is brought closer to sensors if $n' \in Lower(n) \wedge body\_t(r_x) \subset productions(n')$, or it is sent toward a node able to apply it if $n' \in Upper(n) \wedge body\_t(r_x) \not\subset productions(n)$. Incrementally, the rule will converge toward nodes that can no longer propagate it, i.e. $\forall n' \in Lower(n), body\_t(r_x) \not\subset productions(n')$, but that are able to apply it, i.e. $body\_t(r_x) \subset productions(n)$. These are the nodes able to apply the rule that are the closest to the original data producing: propagating the rule lower in the hierarchy is not necessary. Such a node is represented in red on Fig.1. This figure represents the submission of $r_{comfort}$ to the Cloud node by its originator, and its subsequent propagation toward n_5. It is also shown that the node applying $r_{comfort}$ directly provides its originator with its deductions.

Propagating a rule to a lower node does not necessarily entails the inability to apply this rule locally: a node may be both connected to sensors and reasoning nodes. Therefore, after having propagated the rule, a node considers its ability to apply it locally thanks to the types of the properties provided by sensors and lower nodes that are unable to apply the rule themselves. Each node locally adds metadata to the rule to make its status explicit, with the predicate *edr:isRuleActive*. We refer to this property with the functions $mark\_rule\_active(r)$ and $mark\_rule\_inactive(r)$ in Algorithm 1. The activity of the rule is evaluated dynamically when the topology of the network evolves (see Section III-D).

The rule propagation policy supported by EDR is based on the assumption that the **correlation between pieces of data is embedded in the network topology**. IoT data is strongly bound to a spacio-temporal context [23], and the distribution of Fog nodes reflects the distribution of features observed by sensors. From this hypothesis, it can be inferred that the context of a node is a subset of the context of its parent. To illustrate this claim with $r_{comfort}$ previously introduced, it means that if it is possible to apply $r_{comfort}$ with luminosity and temperature observations collected by the same gateway, it is not necessary to compare the same luminosity observations with temperature observations collected elsewhere. IoT data being highly contextual, many applications do not need to reason over a complete KB to get relevant results. EDR is therefore suitable for rules exploiting this context by correlating data sharing an identical context, e.g. the correlation of temperature and luminosity in the context of a single room for $r_{comfort}$. This behavior is adapted to rules supporting deductions for time-sensitive applications, which is the focus of the present contribution, and cannot be applied to aggregation rules, where time series or multiple instances of the same property types are considered. This choice is motivated by the assumption that aggregation rules are more likely to be used in applications supporting long-term reporting and decision support, where the time constraint is not strong, and thus outside the scope of this contribution. To ensure decidability, only DL-safe rules are considered, and EDR is only suitable for stratified rule sets. Cyclic dependencies between rules are not resolved. When a node applies rule r, it is considered as producer of the $head\_t(r)$, and this production information is used for the deployment of any rule r' such as $body\_t(r') \cap head\_t(r) \neq \emptyset$. However, a non stratified rule set where rules r and r' coexist such that $body\_t(r') \subseteq head\_t(r)$ and $body\_t(r) \subseteq head\_t(r')$ cannot be processed successfully by EDR, and neither r nor r' will be propagated or applied.

D. An event-driven algorithm at the core of EDR

Nodes executing the EDR algorithm maintain a coherent view of their neighborhood, and propagate rules with respect to this perception of their environment. The neighborhood of a node is modified when a new node connects or a known node disconnects, and when the productions or consumptions of a node are modified. The main events impacting the exchanges of a node with its neighbors are therefore: when its capabilities are changed (which includes startup and disconnection), when receiving a new rule, and when receiving a new piece of data.

1) When changing capability: Sensors are the primary source of data for the network. The data they produce is collected by their reasoning-enabled parent. When semantic computing-enabled nodes start, they try to connect to their sensors children of which they have a priori knowledge. How nodes discover and gather information about sensors is outside the scope of this paper, as it can be a process tightly related to the underlying technology, or hard-coded in the node KB. Nodes connected to sensors announce the property types they produce to their parent node. As explained in Section III-B3, nodes propagate production information by proxying their children productions. When node n is informed that one of its lower nodes produces a new property type ρ_i, it checks its own productions. If node n already produces ρ_i, the notification is not propagated to $Upper(n)$, supposing it is already aware of the production of ρ_i by n. Otherwise, node n announces itself to $Upper(n)$ as a producer of data of type ρ_i. Similarly, when a sensor or a lower node providing data of type ρ_i to node n disconnects, n announces its updated capabilities if they have been transformed, i.e. if the disconnected node was the sole producer of ρ_i. Announcing both appearance and disappearance of production capabilities ensures the dynamic nature of EDR: nodes' perception of the network topology remains consistent with its evolution over time.

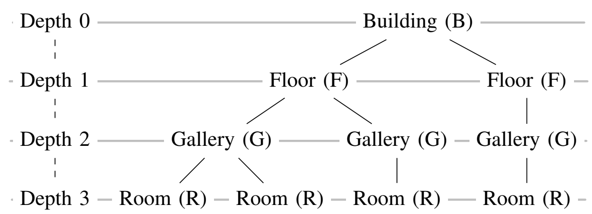
2) When receiving a rule: When node n receives a new rule r, n evaluates whether it can apply r directly, and/or if it should propagate r to some of its children, as described in Algorithm 1. The conditions in which a node locally applies a rule are detailed at line 14. In particular, if node n has both a set of lower nodes producing partially the $body\_t(r)$ (denoted $\mathcal{I}$), and another set of children producing completely $body\_t(r)$ (denoted $\mathcal{C}$), n forwards the rule to $n' \in \mathcal{C}$, but also applies the rule locally by notifying its consumption of the rule body to $n'' \in \mathcal{I}$. In the case where $\mathcal{A} = \bigcup_{n'' \in \mathcal{I}} productions(n'')$ is such that $body\_t(r) \not\subset \mathcal{A}$, node n notifies its interest for the missing rule body elements to its children in $\mathcal{C}$ as detailed l. 25 of Alg. 1. That is why on l. 14, node n determines if it is an aggregator for rule r by checking if $body\_t(r) \subset \mathcal{A}$ and if neither $\mathcal{I}$ and $\mathcal{C}$ are empty.

The moments involving communications between different nodes are at lines 7, 16, 20 and 28 of Alg. 1, where node n respectively propagates a rule (send_rule(n', r)), announces a new production (send_productions($n', \{\delta_j\}$) and advertises for a new interest (send_consumption(n', ρ_i)). The knowledge required to run the algorithm is local to node n. When node n sends rule r to a lower node and does not apply it itself, r is kept in the n's KB. Local metadata is added to rule r in order to keep track of the lower nodes to which it has been transmitted with the predicate *edr:ruleTransmittedTo*.

Algorithm 1 Node n behavior when receiving a new rule

1: **function** PROCESS_NEW_RULE(r)
2: $\quad\mathcal{C} \leftarrow \emptyset$ $\quad\triangleright$ Set of children producing **all** rule's body
3: $\quad\mathcal{I} \leftarrow \emptyset$ $\triangleright$ Set of children producing **some** rule's body
4: $\quad\mathcal{A} \leftarrow \emptyset$ $\quad\triangleright$ Set of types produced by nodes in $\mathcal{I}$
5: $\quad$**for all** $n_l \in Lower(n)$ **do**
6: $\quad\quad$**if** $body\_t(r) \subseteq productions(n_l)$ **then**
7: $\quad\quad\quad$send_rule(n_l, r)
8: $\quad\quad\quad\mathcal{C} \leftarrow \mathcal{C} \cup \{n_l\}$
9: $\quad\quad$**else if** $body\_t(r) \cap productions(n_l) \neq \emptyset$ **then**
10: $\quad\quad\quad\mathcal{I} \leftarrow \mathcal{I} \cup \{n_l\}$
11: $\quad\quad\quad\mathcal{A} \leftarrow \mathcal{A} \cup productions(n_l)$
12: $\quad\quad$**end if**
13: $\quad$**end for**
14: $\quad$**if** $body\_t(r) \subset \mathcal{A} \cup own\_productions(n) \vee \emptyset \notin \{\mathcal{I}, \mathcal{C}\}$ **then**
15: $\quad\quad$mark_active(r)
16: $\quad\quad$send_productions($Upper(n)$, $head\_t(r)$)
17: $\quad\quad$**for all** $\gamma_i \in body\_t(r)$ **do**
18: $\quad\quad\quad$**for all** $n_l \in \mathcal{I}$ **do**
19: $\quad\quad\quad\quad$**if** $\gamma_i \in productions(n_l)$ **then**
20: $\quad\quad\quad\quad\quad\triangleright$ New consum. notified to producers
21: $\quad\quad\quad\quad\quad$send_consumption($n_l$, γ_i)
22: $\quad\quad\quad\quad$**end if**
23: $\quad\quad\quad$**end for**
24: $\quad\quad$**end for**
25: $\quad\quad$**for all** $\gamma_i \in (body\_t(r) \setminus \mathcal{A})$ **do**
26: $\quad\quad\quad$**for all** $n_l \in \mathcal{C}$ **do**
27: $\quad\quad\quad\quad$**if** $\gamma_i \in productions(n_l)$ **then**
28: $\quad\quad\quad\quad\quad$send_consumption($n_l$, γ_i)
29: $\quad\quad\quad\quad$**end if**
30: $\quad\quad\quad$**end for**
31: $\quad\quad$**end for**
32: $\quad$**else**
33: $\quad\quad$mark_inactive(r)
34: $\quad$**end if**
35: **end function**

Fig. 2: Reference simulation architecture



3) When receiving new data: Different kinds of data can be received by node n: raw observations directly produced by a sensor connected to n, and enriched observation or deduction sent to n by node $n_l \in Lower(n)$. If the received observation is raw, node n enriches it by annotating it with an ontology before its processing as a new enriched observation. If the piece of data is either an enriched observation or a deduction, it is directly integrated to its KB and processed. The data, of property type ρ_i, is in the first place sent to $Upper(n)$ if it is a consumer of ρ_i. Then, node n checks if new deductions can be obtained by applying the rules it has marked up as active and whose body includes ρ_i. If the rule body matches the KB of node n, and postconditions of type δ_j are deduced, these deductions are propagated to $Upper(n)$ if it is consumer of δ_j. Since rules are applied on the local KB of node n, there is no impact of data distribution on reasoning complexity. A new reasoning loop is simply applied each time new data is received. The deductions yielded by rule r are also **directly** sent to r's originator(s). Therefore, applications are notified continuously by the nodes as those nodes apply the rules, instead of being notified by a restricted set of central nodes.

IV. Experimentations

A. Motivational use case: living laboratory

To evaluate our approach, we have considered an experimental smart building automation use-case inspired from the real-world example of the ADREAM building[12], an automated smart building and living lab. The use-case building is instrumented with multiple sensors whose observations are transported through a Fog layer constituted of gateways to a Cloud where the data is stored. The building is used for experimental purposes: its behavior is analyzed by multiple applications run by different research teams, and it is also automated by the building administrators. Each research team runs its applications on its own machines (potentially smartphones), and the administrators also monitor and control the building from a dedicated machine, all distinct from the Cloud where data is collected and stored. The applications encompass their logic in rules such as $r_{comfort}$ (as described in Section III-B1), r_{solar}:"With a pyranometry over 430W/m^2, and a temperature over 23.0^{o}C, an energy production under 15000W is abnormal" or $r_{consumption}$: "A difference of over 5 ^{o}C between the requested temperature and the measured temperature in a room will consume a lot of energy". An example energy production analysis application submits to the network rules such as r_{energy} and $r_{consumption}$, while

[12]https://w3id.org/laas-iot

TABLE I: Experimental setup

	RAM	Cores	CPU
Server	32GB	32	3.0GHz
Laptop	16GB	8	2.6GHz
RPi 3	1GB	4	1.4GHz
RPi 2	1GB	4	900MHz

a comfort monitoring application would submit rules such as $r_{comfort}$ and $r_{consumption}$ to make suggestions to the user.

The fact that the building behaviour is automated based on rules motivates their processing as early as possible. The decision process performed by the building administrators based on rule deductions could rely on autonomic computing as explained in our earlier work [21].

B. Experimental setup and implementation

Variations of a reference building's device network topology, described in the use case in Section IV-A and displayed in Fig. 2, are simulated. The root node at depth 0 is the Cloud server while other nodes are Fog nodes. Sensors, not represented on the figure, may be connected under any node. Each sensor pushes a random observation to its parent every two seconds. In order to assess the distributed nature of the algorithm, and its suitability for constrained Fog nodes, the experimental setup includes a Raspberry Pi 2 and a Raspberry Pi 3, a laptop and a server, described in Table I. Each physical machine hosts multiple virtual nodes, composed of an HTTP server, a KB, a SPARQL engine, and a code base[13].

Two aspects of EDR have been evaluated: the validity of our hypothesis, assuming that the distribution of rules increases responsiveness, and the scalability of the proposed approach. Experimental measures also showed that, for each simulation, the number of deductions is consistant between the centralized and the distributed approach: there is no knowledge loss when applying EDR under our assumptions.

To measure the responsiveness of applications enabled by EDR, the **response time between the reception** of the piece of data **and the reception of 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, in order to avoid any impact of the enrichment process on the measure. For instance, if a luminosity observation observed at t_1 and a temperature observation observed at t_2 match $r_{comfort}$ and trigger a deduction received by the originating application at t_3, the response time for this particular deduction will be $t_3 - max(t_1, t_2)$.

C. Impact of the distribution on responsiveness

To measure how distribution impacts responsiveness, four topologies were distinguished, labeled d1 to d4 and further on simply denoted d*. Each of these topologies is constituted of 47 identical nodes, and processes data according to four rules, r1 to r4. The difference between the four d* topologies

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

Fig. 3: d1 topology

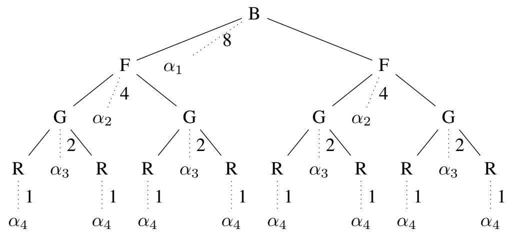


Fig. 4: d2 topology

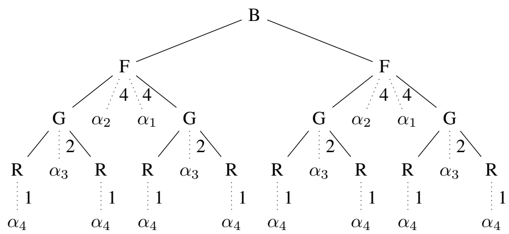
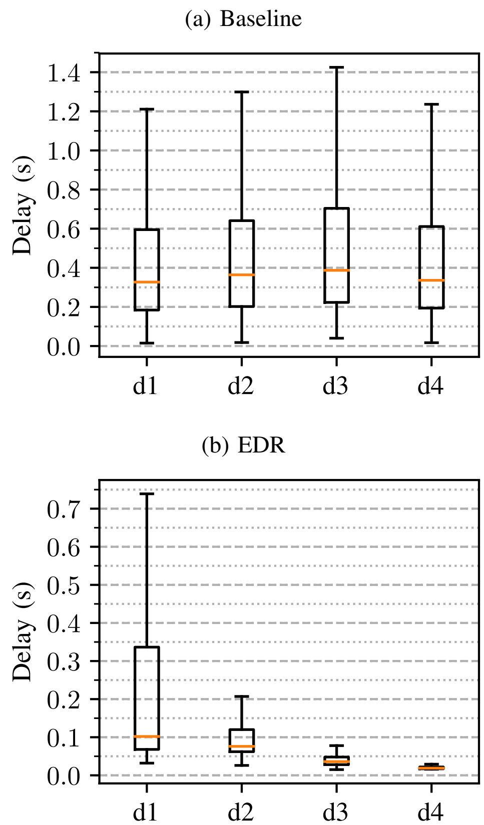


Fig. 5: Performances regarding distribution



is the location of sensors, as depicted in Fig. 3 and 4. Sensors producing data of the type γ_1 are directly attached to the top node in d1, while they are attached to its children in d2. Since $body\_t(r1) = \{\gamma_1, \gamma_4\}$, $r1$ is applied at a maximum depth of 1 in d1, but is propagated to nodes of depth 2 in d2, hence a more decentralized execution is performed in the latter. Rule execution depths are given in Table II: in d4, all sensors are connected to leaf nodes, and the distribution is maximal.

Fig. 5 summarizes the aggregated response time measures for the four rules applied in d* topologies. The data shows

TABLE II: d* topologies

Maximum depth	r1	r2	r3	r4
d1	1	2	3	4
d2	2	2	3	4
d3	3	3	3	4
d4	4	4	4	4

TABLE III: s* topologies

	Nodes
s1	17
s2	27
s3	41
s4	65
s5	78

that the response time remains stable for the four topologies with a centralized approach (Fig. 5a), which is our baseline. Since all computations are performed in the upper node in the centralized case, it is coherent that the distribution of sensor nodes in the network does not impact the response time. With the decentralized approach (Fig. 5b) based on EDR, as expected, the augmentation of the depth at which rules are applied is correlated with the improvement of the response time. Due to the tree-like nature of the network, the deeper a rule can be processed, the more distributed its processing is. Our hypothesis that bringing rules closer to data-producing sensors reduces response time of applications is therefore supported by experimental evidence.

It is clear that not all rules can be processed at the very edge of the network, and that the case depicted in d4 is both purely theoretical and ideal for the EDR approach. This experiment is designed to show that the gain in quality of service is correlated with the possibility to distribute rules, even at a scale where the centralized approach is not clearly outperformed by EDR (as described in Section IV-D).

D. Scalability of the proposed approach

The scalability of the proposed approach is assessed by measuring response time in five topologies, s1 to s5 (denoted s*), with an increasing number of nodes stated in Table III. Nine rules are deployed on s* topologies. Compared to previous d* topologies, the depth at which rules can be applied in s* is constant, as sensors' depth is fixed. The number of nodes is increased by cloning branches in the topologies.

Fig. 6a shows how the response time increases with the number of nodes in the baseline approach. The median response time is stable from s1 to s3, but both the median and the extreme response times increase dramatically in s4 and s5. On Fig. 6b, the response times measured when EDR is applied remains stable from s1 to s4, and the increase observed in s5 is less than the increase observed in the baseline: the two graphics are at very different scales. The experimental evidence supports the claim that rule distribution is scalable.

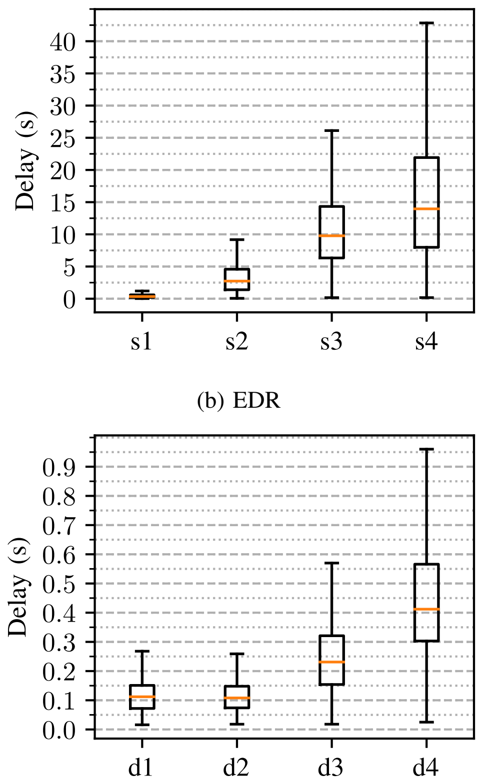
V. Conclusion, perspectives and future work

The presented EDR approach implements a distributed rule processing algorithm that enables the scalable deployment of time-sensitive IoT applications. The approach goes beyond the state-of-the-art work with regard to two aspects. First, it avoids the central control by implementing the distribution of rules on the appropriate Fog nodes, and second, it considers the dynamism of the IoT networks by adapting at runtime the distribution of the rules. Rule-based reasoning is hence performed as close as possible to the sensors that produce the data, and the resulting deductions are directly delivered to the concerned applications. Our approach was evaluated

Fig. 6: Performance regarding scalability

(a) Baseline

(b) EDR



by conducting experiments in a simulated smart building use-case. Experimental results support our claim that first, distributing rule execution improves the responsiveness of the system, and second, the associated algorithm is scalable.

Our results are encouraging for exploring Fog-based semantic-enabled architectures to improve knowledge management in the SWoT, and promote related applications. Enabling the deployment of the Semantic Web stack closer to the constrained devices of the IoT fosters the possibility for reactive time-sensitive interoperable applications, while keeping the data away from centralized Cloud servers. Moreover, shifting from Cloud computing, where the whole produced data is uploaded on remote, potentially third-party servers, to Fog computing, where processing power is brought close to data producers and owners, opens a range of perspectives including privacy-aware distributed semantic reasoning. In our future work, we will investigate how to extend EDR in this direction by considering explicit privacy information in nodes' context. Our algorithm can hence be modified to allow rule propagation in the network for performance and scalability, while data forwarding is constrained by privacy requirements. This constitutes a long term perspective for our work. At a shorter term, we will, first, consider scenarios that go beyond the two extreme configurations of total centralization and distribution of rule processing that are proposed in the present paper. They will be extended in future work to include intermediary, more varying approaches to distribution that we will have to characterize. We will then reconsider our approach in a real deployment scenario at LAAS's ADREAM smart building.

References

[1] J. Kiljander, A. D'elia, F. Morandi, P. Hyttinen, J. Takalo-Mattila, A. Ylisaukko-Oja, J.-P. Soininen, and T. S. Cinotti, "Semantic Interoperability Architecture for Pervasive Computing and Internet of Things," *IEEE Access*, vol. 2, pp. 856–873, 2014.

[2] D. Pfisterer, K. Romer, D. Bimschas, O. Kleine, R. Mietz, C. Truong, H. Hasemann, A. Kröller, M. Pagel, M. Hauswirth, M. Karnstedt, M. Leggieri, A. Passant, and R. Richardson, "SPITFIRE: toward a semantic web of things," *IEEE Communications Magazine*, vol. 49, pp. 40–48, nov 2011.

[3] W. Shi and S. Dustdar, "The promise of edge computing," *Computer*, vol. 49, no. 5, pp. 78–81, 2016.

[4] S. Poslad, S. E. Middleton, F. Chaves, R. Tao, O. Necmioglu, and U. Bugel, "A Semantic IoT Early Warning System for Natural Environment Crisis Management," *IEEE Transactions on Emerging Topics in Computing*, vol. 3, pp. 246–257, jun 2015.

[5] P. Patel, M. Intizar Ali, and A. Sheth, "On Using the Intelligent Edge for IoT Analytics," *IEEE Intelligent Systems*, vol. 32, pp. 64–69, sep 2017.

[6] A. I. Maarala, X. Su, and J. Riekki, "Semantic Reasoning for Context-aware Internet of Things Applications," *IEEE Internet of Things Journal*, 2017.

[7] M. I. Ali, N. Ono, M. Kaysar, Z. U. Shamszaman, T.-l. Pham, F. Gao, K. Griffin, and A. Mileo, "Real-time data analytics and event detection for IoT-enabled communication systems," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 42, pp. 19–37, 2017.

[8] O. B. Sezer, E. Dogdu, and A. M. Ozbayoglu, "Context-aware computing, learning, and big data in internet of things: A survey," *IEEE Internet of Things Journal*, vol. 5, pp. 1–27, 2018.

[9] A. Gyrard, M. Serrano, J. B. Jares, S. K. Datta, and M. I. Ali, "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.

[10] X. Su, P. Li, J. Riekki, X. Liu, J. Kiljander, J.-P. Soininen, C. Prehofer, H. Flores, and Y. Li, "Distribution of Semantic Reasoning on the Edge of Internet of Things," in *IEEE UbiComp*, no. November, p. 79, 2018.

[11] A. Khandelwal, I. Jacobi, and L. Kagal, "Linked rules: Principles for rule reuse on the web," *Lecture Notes in Computer Science*, vol. 6902 LNCS, pp. 108–123, 2011.

[12] A. Sheth, C. Henson, and S. S. Sahoo, "Semantic Sensor Web," *IEEE Internet Computing*, vol. 12, pp. 78–83, jun 2008.

[13] Z. Li, C. H. Chu, W. Yao, and R. a. Behr, "Ontology-driven event detection and indexing in smart spaces," in *IEEE International Conference on Semantic Computing*, pp. 285–292, 2010.

[14] G. Xu, Y. Cao, Y. Ren, X. Li, and Z. Feng, "Network security situation awareness based on semantic ontology and user-defined rules for internet of things," *IEEE Access*, vol. 5, pp. 21046–21056, 2017.

[15] P. Desai, A. Sheth, and P. Anantharam, "Semantic Gateway as a Service architecture for IoT Interoperability," *Networking and Internet Architecture*, p. 16, 2014.

[16] C. E. Kaed, I. Khan, A. V. D. Berg, H. Hossayni, and C. Saint-Marcel, "SRE: semantic rules engine for the industrial internet-of-things gateways," *IEEE Trans. Industrial Informatics*, vol. 14, no. 2, pp. 715–724, 2018.

[17] M. Taneja and A. Davy, "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.

[18] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of Things for Smart Cities," *IEEE Internet of Things Journal*, vol. 1, no. 1, pp. 22–32, 2014.

[19] M. B. Alaya, S. Medjiah, T. Monteil, and K. Drira, "Toward semantic interoperability in oneM2M architecture," *IEEE Communications Magazine*, vol. 53, pp. 35–41, dec 2015.

[20] I. Szilagyi and P. Wira, "Ontologies and Semantic Web for the Internet of Things - a survey," in *IECON*, IEEE, 2016.

[21] N. Seydoux, K. Drira, N. Hernandez, and T. Monteil, "IoT-O, a core-domain IoT ontology to represent connected devices networks," in *EKAW*, 2016.

[22] S. Nikoli, V. Penca, and Z. Konjovi, "Semantic Web Based Architecture for Managing Hardware Heterogeneity in Wireless Sensor Network," *International Journal of Computer Science and Applications*, vol. 8, no. 2, pp. 38–58, 2011.

[23] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, "Context aware computing for the internet of things: A survey," *IEEE Communications Surveys and Tutorials*, vol. 16, pp. 414–454, jan 2014.