Received XX-XX-XXXX; | Revised XX-XX-XXXX; | Accepted XX-XX-XXXX
DOI: xxx/xxxx

LETTER

Reasoning on the edge or in the cloud ?

Nicolas Seydoux[1,2,3] | Khalil Drira[1,3] | Nathalie Hernandez[2] | Thierry Monteil[1,3]

[1]LAAS-CNRS, 7 avenue du Colonel Roche, F-31400 Toulouse, France
[2]IRIT, Maison de la Recherche, Univ. Toulouse Jean Jaurès, 5 allées Antonio Machado, F-31000 Toulouse France
[3]Univ de Toulouse, INSA, LAAS, F-31400, Toulouse, France

Correspondence
LAAS-CNRS Email: name.surname@laas.fr
IRIT Email: name.surname@irit.fr

Abstract

The emergence of the IoT and the Semantic Web of Things is leading to intensive cloud processing executing reasoning rules over large volumes of enriched data. However, the role of the cloud in existing approaches as a central point for both data processing and provisioning reduces scalability and introduces latency. This paper sketches a new approach for rule-based reasoning, enabling distributed rule evaluation on edge nodes and reducing the latency for IoT applications while avoiding the total dependence on a central node. This approach is evaluated in a simulated smart building.

KEYWORDS:
Cloud computing, Edge computing, Rule-based reasoning, Semantic Web of Things

1 | INTRODUCTION

The Semantic Web of Things (SWoT) emerged from the interaction between the Internet of Things (IoT) and the Semantic Web (SW). It is driven by the need for interoperability at the core of the IoT, to which the expressiveness of the SW's formalisms is seen as a solution [1]. Integrating SW technologies into the IoT allows the transformation of raw data into rich, meaningful information that can be distributed with knowledge to applications, thus breaking the vertical silos of domain-specific development. In this configuration, applications both hold the business logic and are consumers of the information produced within the network. However, the SWoT is also challenged by the constrained and dynamic nature of the IoT. This does not comply to the traditional principles and application domains of the SW. Typical IoT deployments follow a hierarchical architecture captured in the Lower, Middle and Upper Node (LMU-N) architectural pattern [2], a multi-tiered IoT architectural pattern where devices connect to cloud servers via intermediate gateways. The entry point to the network for applications is usually a powerful node, e.g. a cloud server offering web services. Data collected by constrained nodes on the edge should therefore be moved upstream toward server nodes so that it can be processed, and sent to relevant applications connected to the cloud.

An application's business logic can be implemented by production rules, where high-level symptoms are inferred from lower-level data, e.g. inferring the high-level event "Such a room is an uncomfortable place" from luminosity and temperature observations. These rules are generally applied in the cloud [3,4], forwarding only the result to applications in order to reduce bandwidth consumption and its processing load. However, this architecture creates a bottleneck, since the data must be concentrated in a single node, the cloud server, for processing. This leads to multiple issues. Firstly, it is not a scalable design, and it is unfit for large deployments [5], such as in smart cities scenarii. Secondly, the cost of semantic reasoning increases rapidly with the size of the knowledge base [6], thus processing all rules on a growing data instance introduces a delay, which can be unacceptable for time-sensitive applications (i.e. e-health, security, emergencies). Thirdly, having a remote central node for reasoning also threatens resiliency since it constitutes a single point of failure.

The paper's purpose is to introduce an approach to data processing suitable for the SWoT, while improving scalability and responsiveness. The proposed algorithm is named Emergent Distributed Reasoning (EDR), and implements a rule-based reasoning approach distributed on the edge by allowing the deployment and execution of rules among nodes. To make this possible, rules and their context must be understandable without ambiguity by the nodes. The present work is focused on production rules, where post-conditions are generated if pre-conditions are valid. Pre-conditions are logical conjunction of atomic tests on values. Aggregations are out of the scope of this work. The remainder of this paper is structured as such: after presenting related works in section 2, the EDR algorithm is described in section 3. Experimental results are then provided in section 4. These results were obtained in a use-case focused on a smart building scenario with the simulation of a smart building network and applications related to home automation. In section 5, after conclusion, future work is sketched.

2 | RELATED WORK

The rules in EDR must be exchangeable, i.e. they can be serialized and interpreted by different nodes. To describe their context, they are enriched with metadata. These characteristics are compliant with the Linked Open Rules principles, an approach based on Linked Rules[7] and extended into Sensor-based Linked Open Rules[3]. Linked Open Rules follow basic principles of Linked Open Data: (1) rules are designated by Unified Resource Identifiers (URIs), (2) these URIs can be dereferenced, (3) they are expressed in standards promoted by the World Wide Web Consortium (W3C), and (4) they can be linked to each other. Regarding the latency introduced by the reasoning, the study performed by Maarala et al.[6] measures how the reduction of the Knowledge Base (KB) size impacts the reasoning time, and shows how the distribution of the processing reduces the bottleneck of centralized reasoning. The impact of the data format, measured in Su et al.[8], is out of the scope of this contribution, as well as the cost of data enrichment. As done in the work of Desai et al.[9], in the proposed approach, data is semantically annotated by the first edge node receiving it, and the time required for this annotation is not considered in the evaluation. Moreover, EDR differs from previous proposals in its focus on the dynamic reconfiguration of rule deployment at runtime involving all the managed system's nodes. Maarala et al.[6] considers a static network and collection of data for reasoning is done on predefined nodes, requiring an a priori knowledge of the topology. In the present approach, the IoT network of sensors producing observations is considered dynamic: sensors and gateways can appear on the network, or disconnect from it, at runtime. The set of rules applied by each node evolves depending on the network topology in order to minimize the processing overhead on each node.

3 | THE EDR ALGORITHM

The motivation for the EDR algorithm is (1) **to increase responsiveness** by reducing the delay between the observation of an event and the notification of relevant applications, (2) **to allow scalable deployment** by basing decisions on knowledge local to each node (3) to **adapt** reasoning to the dynamic nature of the network. Since reasoning time increases with the size of the KB it is performed on, the EDR algorithm aims at applying rules on the smallest possible KB to ensure optimal system responsiveness. A hypothesis for EDR is the hierarchical topology of the device network: data is collected by nodes at the bottom of the hierarchy, while applications connect to the top node. This assumption entails that observations about the physical world are produced by sensors and exchanged upstream in the network architecture, as they are concentrated by middle or cloud nodes using them in inference processes. For a given node, the nodes it is connected to and that are above it in the hierarchy are its upper nodes, and the ones below it are its lower nodes. The purpose of the EDR algorithm is to determine which node is the most suited to apply each production rule, leading to smaller KB and to decisions being taken closer to data sources, therefore reducing the latency introduced by communications. To ensure the scalability of EDR, the knowledge of a node on the network topology is limited to its direct neighborhood, i.e. the nodes it is directly connected to. A node knows the type of data its lower nodes produce. Data is produced by a node if it collects observations from a sensor, or if it obtains it by the application of a rule. Nodes also store in their KB knowledge about rules: the rule itself, and its metadata. The metadata used for EDR are the originator of the rule, i.e. the application that submitted it to the network, as well as the types of the elements in the rule's pre and post conditions (e.g the rule consumes temperature

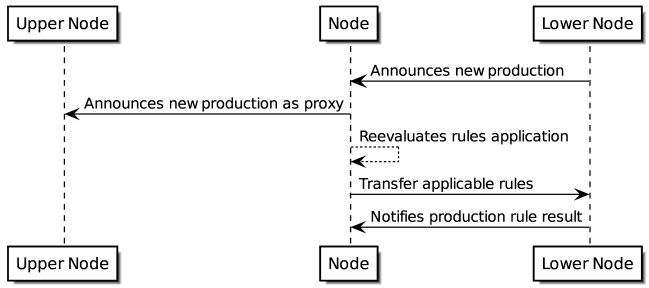

FIGURE 1 New production notification reception

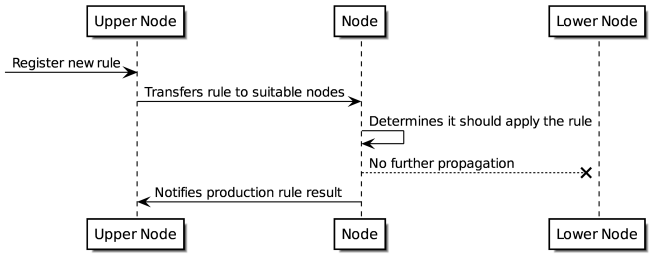

FIGURE 2 New rule reception

observations, and infers comfort, without the actual relation between these concepts expressed in the body of the rule). When a node applies a rule on received observations, it directly notifies the originator of the obtained result.

Let the metadata for the data types of a rule be $r : \alpha_1, ..., \alpha_n \rightarrow \beta_1, ..., \beta_m$, where preconditions are the observation types denoted $\alpha_i, \forall i \in [1, n]$, and $\beta_j, \forall j \in [1, m]$ the postconditions corresponding to the types of the generated data. In order for r to allow inference, it must be applied by a node which is the common ancestor to producers of $\alpha_i \forall i \in [1, n]$. Because of the tree-like architecture of the network, the first common ancestor is the node that will be the common ancestor of the fewest data sources, and therefore the least bottleneck to the reasoning process of the the common ancestors. The purpose of EDR is to transfer each rule to the lowest node possible in the architecture, each node propagating the rule based on local knowledge of its neighborhood. Moreover, due to the dynamic nature of IoT networks, the decisions taken by the EDR algorithm can evolve over time if a better solution emerges as the knowledge of a node on the topology evolves.

All the reasoning nodes of the network (sensors are not considered reasoning nodes) have the same behaviour, composed of four main events: when it starts, when it is notified of the evolution of the topology of the surrounding network, when it receives a new rule, and when it receives a new observation. These functions are executed by the node itself, which interacts only with its own KB in order to ensure the decision is taken locally. The behaviour of the node for each event is described in the paragraphs below.

A. On startup and disconnection

When it starts, a node notifies its upper node(s) about the type of observation it produces, based on the knowledge about the sensors it is directly connected to. The discovery and introspection methods are out of the scope of this paper. Similarly, if a sensor or a node disconnects, it notifies its upper node about the interruption of its productions.

B. Process new notification

To maintain a coherent representation of the network, nodes send and propagate semantically enriched notifications when they connect, disconnect or move in the graph, and when their capabilities evolve: production of a new type of observation thanks to a rule application or to the connection of a sensor as a lower node (represented on fig. 1), as well as interruption of a production (disconnection of a sensor among its lower nodes). The notifications sent between nodes ("Announces..." on fig. 1) contain a semantic description of the types of data they produce. Each time a notification is received, the knowledge of the target node on its surroundings (e.g. the observation types produced by its lower nodes) is updated. To ensure a coherence between the KB of the different nodes, new production capability notifications are propagated upstream by nodes upon reception. As EDR relies on a **local** representation of the network for each node, when generating the notification to propagate, the node acts as a proxy [10], and claims to be the producer of the new observation type.

The update of the target node is propagated to its upper nodes if its productions have been modified. When a node is notified of an evolution of the network topology, it reevaluates whether or not it is an aggregator for the rules it holds as described in the next paragraph. On the one hand, the node can decide to send the rule to a child node if this child is now able to apply the rule, which is the case in fig. 1. On the other hand, the node can send the rule back to its direct upper node if it is no longer able to apply it.

C. Process new rule

The propagation of a rule from a node to another is illustrated with the diagram on fig. 2. When receiving a rule, a node evaluates if it is applicable by himself. If the node is a producer of all the rule preconditions, or if it is an **aggregator** for these preconditions, it marks the rule as applicable. A node is an aggregator if it has a set of child nodes that produce together all the preconditions for a rule, but none of them produces them entirely on its own. If the rule is applicable, the node notifies its upper nodes that it now produces the post conditions of the rule. In the example on fig. 2, the central node is an aggregator for the rule: it does not transfer it to its lower nodes. A node also forwards a newly received rule to any of its child nodes producing all of the rule's preconditions, because these child nodes are in this case common ancestors closer to the producers of the rule body elements.

D. Process new observation

When a node receives a new observation from another node, it checks if new deductions can be obtained by applying the rules this node holds. If the rule matches and postconditions are deduced, the deductions are propagated to upper nodes. In any case (deduction or not), the observation is also sent to the upper node as it may hold a rule consuming it. Moreover, the deductions of the rule are also sent to its originator (the application that declared the rule). The endpoint of the originator is described as a metadata of the rule, so any node applying the rule can directly notify the application. Therefore, applications are notified continuously by the nodes as they apply the rules, instead of being notified by a restricted set of central nodes.

4 | EXPERIMENTAL RESULTS

A first implementation of the EDR algorithm is available freely[1]. The first simulations feature three applications consuming data generated randomly by an IoT network emulated by communicating processes organised in a hierarchical topology summed up in Figure 3. The cardinalities stated on the edges of the figure show how the reference topology can be extended by adding multiple "Floor", or "Gallery" or "Room" nodes. The **depth** of a node is the number of hops between this node and the root, so for instance a "Gallery" (cf fig. 3 topology) node is at a depth of 2, and a "Room" (cf fig. 3) node at a depth of 3. Each application expresses business rules, i.e. "If the temperature and the luminosity of a room are below a certain threshold, this room is considered uncomfortable for work" or "If the difference between the measured temperature and the target temperature is above a certain threshold, reaching the required temperature is energy-consuming". Five rules are used for simulation, available with the source code[1]. These rules are distributed using the EDR algorithm in order to be executed by the deepest possible node.

With these simulations, the moment when all the observations required for the deduction are available, and the time the deduction is actually made by the relevant node, can be compared to compute delays. The node which makes the deduction sends the result directly to the application that registered the associated rule. The time required to transfer the said deduction to the application being considered the same regardless of the node, this does not impact measures. The behaviour of the proposed algorithm has been studied according to two parameters of the networks: topologies where rules can be applied at different depths (see tab. 1), and topologies with an increasing number of nodes (see tab. 2). These network topologies are variations of the reference topology represented on fig. 3, listed in table 2. The evaluations were run on a 16-core server with 32GB of RAM, and captured on 12-minute data samples, with an observation produced by each sensor every 5 seconds. The results show a reduction in the delay between a scenario where EDR is applied and a centralized cloud reasoning baseline (for the same topology).

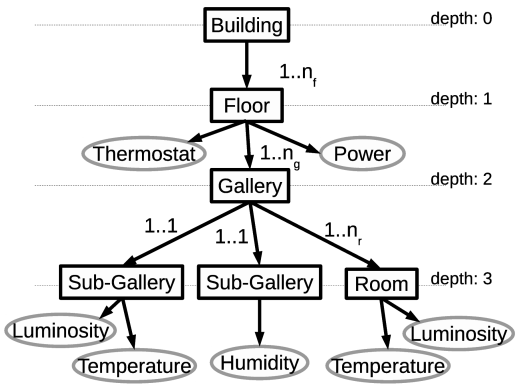

FIGURE 3 Overview of the reference topology

[1] https://framagit.org/nseydoux/edr The code is actively developed, and might implement more features than described in the present paper.

TABLE 1 Distribution measurement topologies

	rule 1	rule 2	rule 3	rule 5	rule 6	Max depth	Node count
Topo. d1	0	0	0	0	0	3	49
Topo. d2	2	3	3	2	1	3	49
Topo. d3	3	3	3	2	1	3	46
Topo. d4	3	3	3	3	2	3	50
Topo. d5	3	3	3	3	3	3	56

TABLE 2 Scalability measurement topologies

	Node count	n_f	n_g	n_r
Topo. s1	13	1	1	1
Topo. s2	34	1	2	3
Topo. s3	43	2	2	1
Topo. s4	67	2	3	3

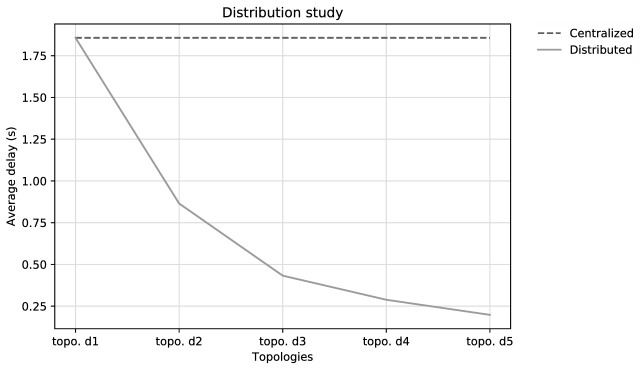

FIGURE 5 Distribution simulation results

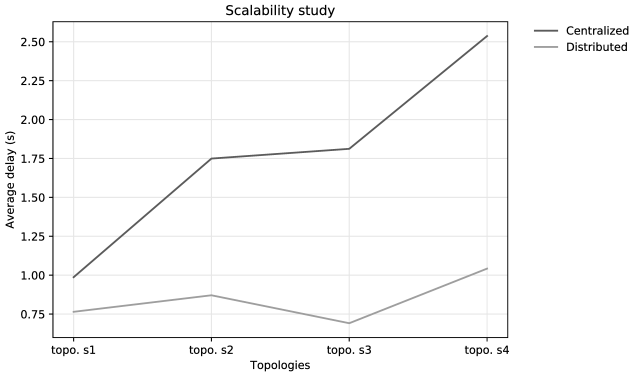

FIGURE 6 Scalability simulation results

The scalability study is summed up in figure 6. It represents the evolution of the delay according to 4 topologies of increasing number of nodes detailed in tab 2.The impact on the delay of the number of nodes is much more important in the centralized case than in the distributed case, which supports the claim that EDR is an approach more scalable than the baseline.

To measure the impact of the distribution of rules, simulations were conducted on several topologies, listed in tab. 1. All these distributions are of a comparable size, but the location of the sensors (represented with grey circles in fig. 3) is modified in order for the rules to be applicable at different depths. Topology d1 is centralized. For this reason, all rules are executed by the root node at a depth of 0. For other topologies, tab. 1 describes the depth at which rules can be applied. Fig. 4 shows the instantiation of the d3 topology. Fig. 5 shows that in topologies where rules can be executed by deeper nodes, i.e. in a more distributed manner, a more responsive behaviour of applications is measured, validating the core hypothesis of EDR.

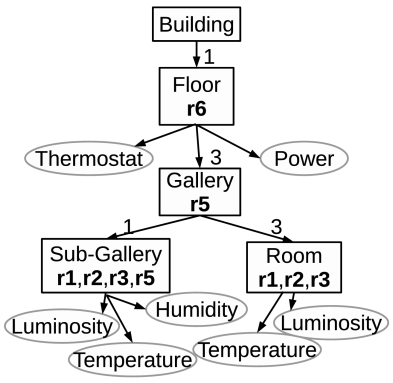

FIGURE 4 Instantiation of d3

Two factors impact the reduction of delays observed in the experiments: the reduction of the number of message exchanges between the node making the observation and the node making the deduction, and the reduction of the reasoning time induced by the reduction of the KB size . The average time for a message exchange in the experimental setup was 17ms, therefore executing rules at a deeper level saves this trip time whenever a conclusion can be made locally. Moreover, reducing the size of the knowledge base by performing local computing enhances reasoning performances, with reasoning times reduced from 3 to 10 times when switching from centralized to distributed reasoning.

5 | CONCLUSION AND FUTURE WORKS

The contribution sketched in this paper is EDR, a distributed rule-based reasoning algorithm aiming at offloading computation from the cloud to the edge, enabling SWoT deployment for IoT applications. The first results of the application of the EDR algorithms are encouraging, and support the claim that this algorithm increases responsiveness

of IoT applications, while its distributed nature ensures its scalability. Even if these first results are very promising, it should be noted that they were obtained on simulations that fail to capture some aspects of the problem. For instance, this first version of the algorithm is oblivious of the disparity of processing powers between nodes. Edge nodes, even if they reason on smaller knowledge bases, do not have the same processing power as cloud nodes, which is not in the scope of this paper but will be considered in future work.

Further evaluations will also be performed on real datasets considering on a larger variety of topologies to assess the veracity of the hypothesis on hierarchical deployments. Rules requiring aggregation will also be considered. The EDR algorithm, by distributing reasoning on the edge and connecting gateways to applications directly, reduces the dependency of applications on the central node they register to initially which should increase the resiliency of the network. The behaviour of the algorithm deployed in a faulty network will also be part of future work.

References

1. Sheth Amit, Henson Cory, Sahoo Satya S. Semantic Sensor Web. *IEEE Internet Computing.* 2008;12(4):78–83.

2. Seydoux Nicolas, Drira Khalil, Hernandez Nathalie, Monteil Thierry. Capturing the contributions of the semantic web to the IoT: a unifying vision (extended abstract). *Semantic Web technologies for the Internet of Things.* 2017;.

3. Gyrard Amelie, Serrano Martin, Jares Joao Bosco, Datta Soumya Kanti, Ali Muhammad Intizar. Sensor-based Linked Open Rules (S-LOR): An Automated Rule Discovery Approach for IoT Applications and its use in Smart Cities. *Proceedings of the 26th International Conference on World Wide Web Companion.* 2017;:1153–1159.

4. Meng Z., Lu J.. A Rule-based Service Customization Strategy for Smart Home Context-Aware Automation. *IEEE Transactions on Mobile Computing.* 2016;15(3):558–571.

5. Wang Wei, De Suparna, Zhou Yuchao, Huang Xin, Moessner Klaus. Distributed Sensor Data Computing in Smart City Applications. *Proceedings of WoWMoM 2017.* 2017;.

6. Maarala Altti Ilari, Su Xiang, Riekki Jukka. Semantic Reasoning for Context-aware Internet of Things Applications. *IEEE Internet of Things Journal.* 2017;.

7. Khandelwal Ankesh, Jacobi Ian, Kagal Lalana. Linked rules: Principles for rule reuse on the web. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics).* 2011;6902 LNCS:108–123.

8. Su Xiang, Riekki Jukka, Nurminen Jukka K., Nieminen Johanna, Koskimies Markus. Adding semantics to internet of things. *Concurrency Computation.* 2015;27(8).

9. Desai Pratikkumar, Sheth Amit, Anantharam Pramod. Semantic Gateway as a Service architecture for IoT Interoperability. *Networking and Internet Architecture.* 2014;:16.

10. Nikoli Siniša, Penca Valentin, Konjovi Zora. Semantic Web Based Architecture for Managing Hardware Heterogeneity in Wireless Sensor Network. *International Journal of Computer Science and Applications.* 2011;8(2):38–58.

How to cite this article: Seydoux N., Drira K., Hernandez N., and Monteil T. (XXXX), An emergent distributed approach to rule-based continuous reasoning in an IoT networks, *Internet Technology Letters*, *XXX;XX:X–X.*