LATTICE: A Framework for Optimizing IoT System Configurations at the Edge

Valérie Issarny, Benjamin Billet, Georgios Bouloukakis, Daniela Florescu, Cristian Toma

LATTICE: A Framework for Optimizing IoT System Configurations at the Edge

Valérie Issarny*, Benjamin Billet, Georgios Bouloukakis*†, Daniela Florescu‡, Cristian Toma§
*Inria, Paris, France
†University of California, Irvine, CA, USA
‡ARICA, Palo Alto, CA, USA
§Kalera Inc., Orlando, FL, USA

***Abstract*—The Internet of Things is expected to contribute to a "smarter world" by connecting the physical to the virtual, i.e., enabling advanced knowledge engineering over the big data gathered about the physical world. However, such a promise comes along with high resource consumption, spanning the network, storage and computational resources, not to mention possible security and privacy threats. As a result, it tends to be admitted that the IoT smartness will not be accommodated at scale by a centralized cloud-based approach. Instead, the deployment of IoT systems needs to leverage a highly distributed system architecture, which optimizes the distribution of the computation –from the edge to the cloud– according to the unique business requirements in terms of financial cost, latency, availability, *etc*. Toward that goal, this paper introduces the LATTICE framework, which aims at taming the complexity of configuring edge-based IoT systems. LATTICE builds upon ontologies that have proven useful to characterize the constituents of IoT systems in the required domain-specific way. However, LATTICE also revisits the exploitation of ontologies –i.e., the formal description of the real world, spanning the physical and cyber entities– across the development life-cycle of the IoT systems. As a first evidence, this paper introduces an automated approach to the optimization of IoT system configurations at the edge, provided the ontological description of the target IoT system.**

***Keywords*-Internet of Things; Edge computing; Ontologies; Optimization.**

I. INTRODUCTION

The *Internet of Things* (IoT) holds the promise of blending the physical and virtual worlds. Connected objects enable the virtual world to aggregate knowledge about the physical world across time and space, and to leverage this knowledge to act upon the physical world. In other words, we expect our physical world to become "*digitally smarter*" as illustrated by the addition of the smart adjective to most physical environments. For instance, the IoT enables *smarter cities* to monitor and act upon the urban environment at an unprecedented scale. The foreseen positive impact is the ability to overcome the critical issues that the large cities are facing like the increased consumption of natural resources, exposure of the population to environmental pollution, urban traffic, *etc.* [1].

There is no doubt that technologies already offer a wealth of solutions for the society to benefit from the IoT: Wireless Sensor & Actuator Networks (WSAN) have taught us how to deploy wireless, resource-constrained devices in a variety of places –from body to harsh environments– [2]–[4]; Cloud computing brings the computational and storage capacity needed to collect and analyze the big data that the IoT produces [5]–[7]; Machine learning and data analytics enable us to learn about the physical environment and even anticipate the future from the gathered observations [8]–[10]. The list of relevant technologies is much longer and keep expanding, and we already witness numerous successful deployments of IoT systems in many vertical sectors. Still, the deployment of IoT solutions is far from being a norm in the various verticals that it may support. Obviously, there are economic considerations related to the definition of the business models associated with the cost-effectiveness of IoT solutions, which are beyond the scope of distributed computing. However, and this is the focus of this paper, distributed computing research has a major role to play in *democratizing the IoT* so that IoT systems may easily be deployed and operated for the social good –in the most cost-effective way.

The starting point of our research is that the development of IoT systems is hard despite the existing enabling technologies. IoT systems are among today's most challenging distributed computing systems because they require extensive expertise in both the vertical domain they support (e.g., consider the complexity of the urban environment and even the specific example of urban pollution monitoring [11]), and the digital sciences and technologies that they must compose (cf. the above list of underlying technologies). Bringing together expertise in the digital with some vertical domain is not a new concern. Software system engineering has in particular brought significant enablers through requirements engineering methods and tools [12], as well as domain-specific languages [13]. Ontological engineering has also proven a valuable tool to address the requirements of domain-specific systems [14]. However, the development of IoT systems further requires dealing with the deployment of the "Things" in the physical area that they instrument. And, this deployment potentially involves a large number of highly heterogeneous things that produce 24/7 data streams. In a nutshell, the challenge facing the development of IoT systems is that of highly complex IT systems, while further requiring very specialized domain knowledge. We argue that the formal description of the real world, i.e., using ontologies, enable taming such a complexity in the required domain-specific way, but that a major re-

think is needed regarding how ontologies are exploited in the development of IoT systems.

The use of ontologies in IoT systems is not new; it has been stressed as a major enabler since the early days of the IoT. Precisely, ontologies primarily serve two related, yet complementary, purposes in the deployment of IoT systems: (i) reasoning about the gathered data by characterizing the concepts of the specific physical domain under scrutiny [15], [16], and (ii) interoperability by characterizing the connected things through the relevant metadata [17], [18]. However, the complexity of IoT systems is not only due to their heterogeneity, their scale and uncertainty are two other complexity factors. Still, we argue that ontologies may serve managing the overall complexity of distributed IoT systems at all the phases of the system development, from architecture design to the implementation of the system components:

- *Understanding the vertical domain:* By their very definition, ontologies characterize the various concepts of a given domain, and their relation [19], [20].
- *Facing heterogeneity*: We have already stressed that ontologies have been extensively used to enable interoperability within IoT systems. However, we emphasize that ontologies allow overcoming heterogeneity across the distributed system stack [21]. That is, ontologies enable reasoning about the semantics of distributed protocols and further synthesizing mediators that reconcile their functional mismatches [22].
- *Facing scale*: The large scale of the IoT manifests itself with respect to the number of things, and the volume of data that is generated. Dealing with the former is about abstracting the problem space, while the physical domain is often structured around repetitive concepts (e.g., consider a city map). Hence, ontologies provide us with the right engineering tool to define the domains' abstractions. Dealing with the high data volume of the IoT primarily relies on distributed system paradigms for which edge/fog computing is becoming an appealing solution [23], [24]. Still, and this is the focus of our paper, ontologies enable tackling the optimization of the system configurations at the edge.
- *Facing uncertainty*: Uncertainty is a built-in features of the IoT, from the one of the availability of Things to the one of the confidence (accuracy) of the provided knowledge. The semantics of things and of their observations enable reasoning about their composition, correction, and replacement so as to overcome the uncertainty that arises [17].

Democratizing the deployment of IoT systems by revisiting the exploitation of ontologies –i.e., leveraging the formal description of the real world, spanning the physical and cyber entities– across the development life-cycle is a long term research agenda. As a first evidence, this paper focuses on the role that ontologies may play to optimize the configuration of IoT systems at the edge, thereby addressing the system's scale in a domain-specific way. After an overview of edge-based IoT systems within Section II, the paper makes the following contributions:

1) We introduce the LATTICE ontology-based framework for the development of IoT systems, which generate an optimized configuration of the systems provided the ontologies associated with their vertical domains (Section III).
2) We detail the ontologies that are at the core of the LATTICE framework, and derive from the state of the art IoT system models (Section IV).
3) We show how to leverage the IoT system ontologies to synthesize the optimization problem associated with the placement of the computational nodes and the allocation of tasks, at the edge, and further generate the corresponding optimizer code (Section V).

Finally, we conclude with areas for future work in Section VI.

II. BACKGROUND

Initially, the architecture of IoT systems were largely cloud-based. That is, the systems were structured around a Wireless Sensor & Actuator Network to sense and act upon the physical environment, while the sensor data are transferred toward the cloud for analysis and controlling back the actuators. However, with the increasing very large scale –either happening or foreseen– of the IoT, an emerging trend in IoT applications is to move the computation closer to the source of the data [25]. The motivations are many-fold, among which: (i) the scale of the systems in terms of the number of sensors and the volume of data leads to massive resource consumption from the network up to the computational resources, (ii) many of the IoT applications involve timeliness and even safety-critical requirements that a centralized architecture cannot guarantee at scale, and (iii) the increasing concern for privacy and security challenges the adequacy of aggregating all the knowledge at the central cloud. Following, since the last few years, there is extensive research and development on supporting the deployment of edge/fog computing as part of IoT systems.

The literature is rich of definitions for fog and edge computing [26], including by industrial consortia [27], [28]. In particular, the boundary between fog and edge computing is tiny and the two terms are sometimes used interchangeably. In our work, we tend to consider the two as similar and referring to bringing computing and storage resources closer to the devices (i.e., at the edge), while fog computing additionally stresses the virtualization of the infrastructure.

Research on easing the exploitation of edge computing within IoT applications then ranges from the study of the supporting programming models and framework, to the study of the edge placement. Regarding the definition of programming models for IoT applications that get deployed over edge nodes, most solutions adopt a distributed data flow approach where the development of the application subdivides into the implementation of the node functions, and the composition of those functions [29], [30]. This further leads some work to focus on the development of libraries for the implementation

of advanced features at the edge nodes like online learning [31]. The deployment of IoT systems featuring edge nodes also requires dealing with the placement of the nodes. This is addressed as an optimization problem, where solutions differ depending on whether they focus on the placement of the IoT services on *a priori* deployed edge nodes [32], or on the deployment of the edge nodes themselves [33], [34].

In accordance with the above evolution of IoT systems, initial industry solutions for the IoT were largely cloud-based with the processing addressing: (a) operational decisions (i.e., implementing the business rules), and (b) the feedback loop via machine learning (i.e., automatically inferring the business rules). Most efforts focused on the second aspect (i.e., machine learning). Solutions include for instance Siemens MindSphere[1] and the IBM Watson IoT platform[2]. So as to address the limitations arising with the cloud-based solutions (networking cost, lack of security, increased latency, reduced reliability and availability), large cloud corporations have invested efforts into edge-based IoT. Most of the efforts are dedicated to facilitating the writing of software that implement the business decisions on the CPU closer to the sensors and actuators. Technically, this translates into better support for streaming and data analytics at the edge. Related solutions includes the AWS green grass[3] and Azure IoT Central[4]. In a complementary way, solutions providing the horizontal capabilities required at the edge are emerging like Intel CPU Oat[5], Cisco Kinetic[6], Mulesoft data integration[7], *etc*.

However, despite the significant effort devoted developing edge-based IoT systems, little consideration has been given to the problem of designing the global architecture for cloud and edge processing, where part of the computations are done at the edge, and part on the cloud. In general, very little effort is dedicated to help users find a global optimal solution based on their unique business requirements in terms of financial and resource costs, latency, availability, etc. One of the very few offerings in the industry in this area is PTC Thingworx[8]. Thingworx offers a tool helping users to design a global architecture, yet all the decisions are still taken manually, despite the complexity of the problem. Code is also written by hand and not automatically generated, which is known to be highly inefficient and error-prone. On the other hand, the optimization of node placement has been extensively studied in the networking domain with the introduction of diverse heuristics depending on the target problem (e.g., [33]), which may be leveraged for the configuration of edge-based IoT systems.

In this paper, we show that the optimization problem may be generated from the ontologies associated with the definition of:

[1]https://siemens.mindsphere.io
[2]https://www.ibm.com/cloud/watson-iot-platform
[3]https://aws.amazon.com/en/greengrass
[4]https://azure.microsoft.com/en/services/iot-central
[5]https://www.intel.com/content/www/us/en/wireless-network/accel-technology.html
[6]https://www.cisco.com/c/en/us/solutions/internet-of-things/iot-kinetic.html
[7]https://www.mulesoft.com/integration-solutions/dataweave-integration
[8]https://www.ptc.com/en/products/iot

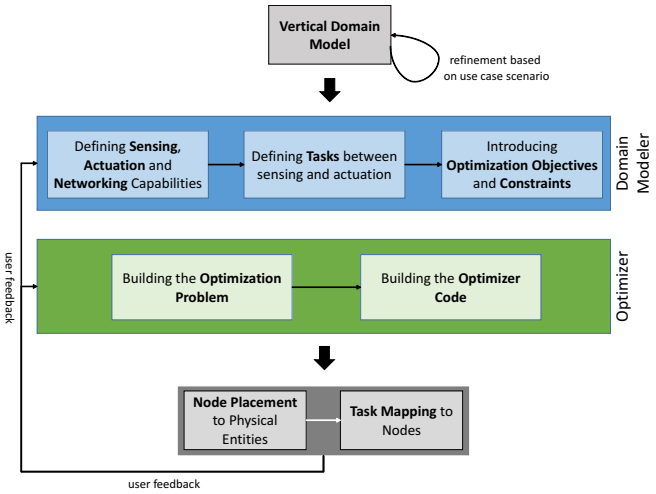

Fig. 1. The LATTICE Node Placement and Task Mapping process.

the physical entities, the data flow among the related sensors and actuators, and the optimization requirement for the system. The vision is that starting from the ontologies characterizing the –vertical and digital– domain space, we are able to address the overall development, from architecture design to node implementation, of edge-based IoT systems. We specifically introduce the design of the supporting LATTICE framework, together with how it enables generating the optimizer code for a given IoT configuration.

III. THE LATTICE FRAMEWORK

Figure 1 depicts the key elements of the LATTICE framework:

- The framework takes as input the ontologies associated with the target use case, which relate to (see § IV for detail): (i) the physical entities, (ii) the cyber entities used for sensing and actuation, and (iii) the constraints and optimization parameters for the IoT system. The ontologies leverage as much as possible existing standards (see [18] for a survey). In particular, there exist various ontology standards for characterizing application domains as well as IoT networks.
- Provided the ontologies, the *domain modeler* instantiates the initial IoT configuration, i.e., the modeler defines: (i) the sensing, actuation and networking capabilities that are deployed in the physical space, (ii) the processing tasks that need to be deployed between the sensors and actuators, and (iii) the optimization objectives and constraints for the system. We note that the work of the modeler may be quite straightforward considering the repetitive nature of most physical spaces. For instance, the operation of a building may be addressed at the level of a floor (the IoT system being the same at each floor), and the design of the IoT system at the floor level may even be simplified by abstracting all the similar rooms as a single room –at least from the modeling perspective.

Fig. 2. A hydroponics room.

- Once the model of the IoT system is available, the framework builds a constrained optimization problem in order to compute the best solution for computational node placement and task allocation (see § V for detail).

The two next sections detail the LATTICE core where we take the example of a hydroponics system [35] for illustration. In a hydroponics room, the vegetables are grown in nutrient solutions (see Figure 2). The supporting IoT system must thus monitor and control various physical variables, e.g., temperature, humidity, CO2, air flow, lighting, and fertilizer concentration, balance, and pH [36], [37].

IV. CHARACTERIZING THE PROBLEM SPACE USING ONTOLOGIES

Ontological engineering is about abstracting the knowledge of domains as a set of concepts and relationships among those concepts. This is enabled through the formal specification of individuals (instances of objects), classes, attributes, *etc*. In particular, an *object property* specifies a relation between class instances, while a *data property* specifies a relation between an instance and a simple data value (string, integer, *etc*.).

Ontological engineering has proven useful for various automated applications (AI, knowledge engineering over the web, *etc*.) as it eases refining and reasoning about such knowledge. The LATTICE framework specifically leverages ontologies to formally characterize –both the domain-specific and the digital– constituents of the IoT systems, which subdivide into:

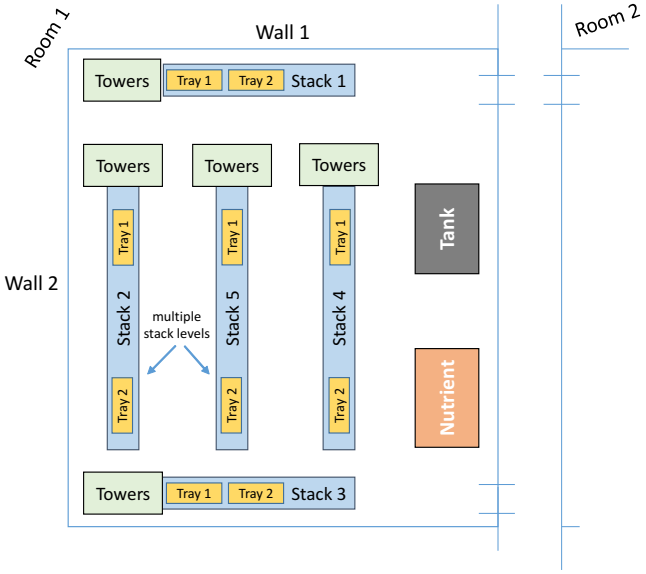

Fig. 3. The physical entities of the hydroponics room.

1) The *physical entities* (e.g., the trays of the hydroponics room) defining the *physical space* (e.g., a hydroponics room) that the IoT system monitors and controls;
2) The *cyber entities* that define the network of things underlying the system (i.e., sensors, actuators, computational nodes and networks);
3) The *data flows* across the cyber entities to actually monitor and control the space that the system implements.

As we sketch in Section I, leveraging the ontology-based formalization of the *physical* and *cyber* entities composing an IoT system is a classical approach to overcome the heterogeneity of the Things. As such, and although this is beyond the scope of this paper, the LATTICE framework may exploit existing ontologies [18], while the characterization of data flows follows from state of the art programming models associated with edge-based IoT systems [29].

The focus of this paper is to show how we exploit the ontology-encoded knowledge about IoT systems to optimize the configuration of the system at the edge –i.e., to optimize the placement of the computational nodes that host the tasks collecting sensor data and controlling the actuators. We first outline the associated ontological engineering for which we use the hydroponics scenario for illustration, providing an excerpt of the associated ontology definition.

A. Physical & Cyber Entities

The LATTICE framework takes as input ontologies characterizing the *physical* and *cyber* entities. The needed ontologies may be either already available or introduced by the domain modeler. Ultimately, the domain modeler leverages the ontology characterizing the physical space that the target IoT system monitors and controls, i.e., the hydroponics room in our illustrative example.

Figure 3 depicts the physical entities of the hydroponics room of Figure 2: the room is made up of multiple stacks; each stack consists of multiple levels; and, at each level, there are multiple *trays* with plants. Additionally, each stack is connected to multiple *towers* that control the water and nutrient levels within each tray. Finally, the hydroponics room has one *tank* and one *nutrient* controllers that adjust the nutrient and water levels of the whole room.

Figure 4 illustrates the corresponding ontology formalization, starting from the *Room* concept. The *hydroponics physical entities* (e.g., `NutrientController`, `Stack`) are introduced through a number of object properties (e.g., `hasNutCntr`, `hasStack`). The "*deployment*" of the *cyber entities* (i.e., sensing and actuation) across the physical entities is also specified using object properties (e.g., `hasNutSensing`, `hasNutActuation`); they deal with controlling the nutrient levels within the room.

To enable the data exchange between the cyber entities, the access network (e.g., WiFi or ZigBee) is defined using the `Network` class. Additionally, we define data properties for representing network characteristics, such as `netRange`, `dataRate` and `latency`. Without loss of generality, we assume that the network is set at the room level (see `hasNetwork`). However, a domain modeler may refine the current ontology by introducing different types of networks at different physical entities. For example, the cyber entities of `TankController` may interact with each other through WiFi, while the cyber entities of `NutrientController` may interact through ZigBee. Such high-level network classes can also be extended to represent actual networks, e.g., a specific ZigBee network could link the `HydroponicsRoom` to the `NutrientController` and the `TankController` while another ZigBee network could link the `HydroponicsRoom` to the `Stack`.

Last but not least, and as depicted in Figure 5, a cyber entity (e.g., `NutrientSensing`) is a super class of sensor and actuator classes of the nutrient controller (`HeartbeatWaterTemp`, `AcidPump`, etc). To characterize the data generated within the IoT system, we define data properties relations related to *accuracy*, *data rates*, *data sizes* and *data formats* for sensors as well as *data rates* and *data formats* for actuators. Note that such characteristics (and possibly others) may leverage well-known, third-party, ontologies for the IoT such as SOSA/SSN [38].

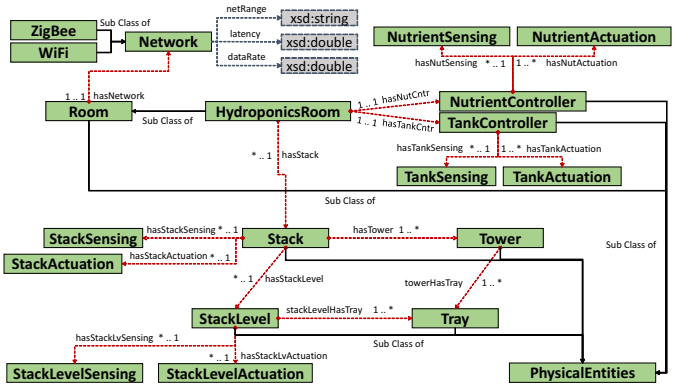

Fig. 4. Formalizing the physical and cyber entities of the hydroponics room.

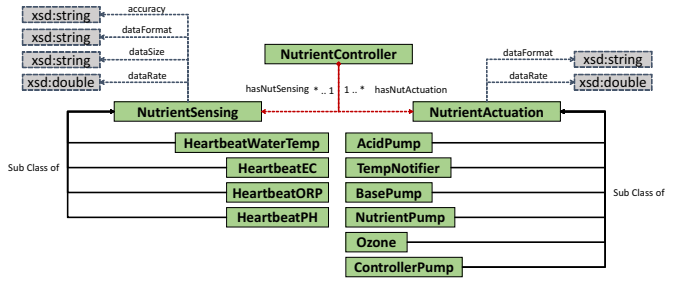

Fig. 5. Data properties associated with the cyber entities.

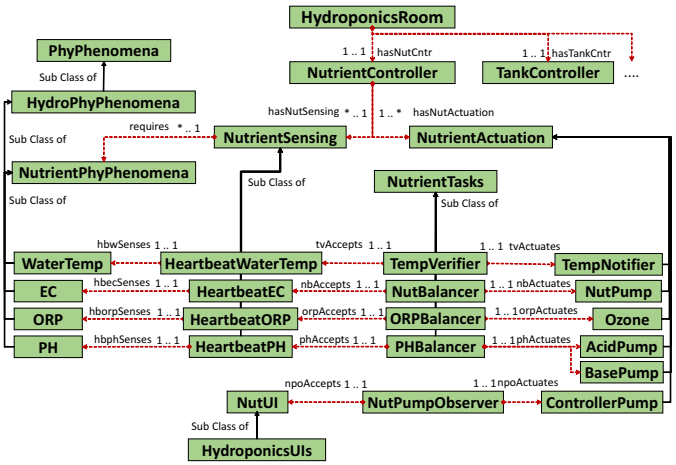

Fig. 6. Nutrient controller data flows.

B. Data Flows

To define the data flows associated with the physical entities, we first define the physical phenomena that must be monitored. Then, we define the tasks that process these phenomena to control the physical entities. More specifically, one or more sensors *sense* some physical phenomena and each task *accepts* the sensed data to *process* them and then *actuate* one or more actuators. For example, for a given physical entity (e.g., the `NutrientController`) the domain modeler has to define the associated physical phenomena (e.g., the nutrient controller's physical phenomena `NutrientPhyPhenomena`), as well as the tasks (`NutrientTasks`) processing the data flows coming from the related sensors for possible actuation (see Figure 6). For instance, as depicted on Figure 6, the `HeartbeatWaterTemp` sensor senses the `WaterTemp` physical phenomenon, while the task `TempVerifier` accepts `HeartbeatWaterTemp` raw data to process them and actuate the `TempNotifier` actuator. To enable human interactions, the `NutPumpObserver` task accepts data from the `NutUI` in order to control the pump of the room (`ControllerPump`). Similarly, the domain modeler can specify data flows between sensors/UIs and actuators in the

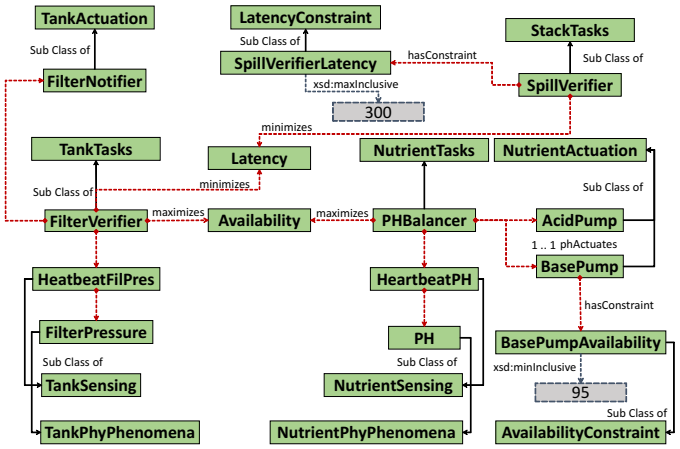

Fig. 7. Example of minimization and maximization of properties.

$\mathcal{N}$, $\mathcal{T}$, $\mathcal{A}$, $\mathcal{S}$, $\mathcal{P}$, $\mathcal{L}$	Sets of nodes, tasks, actuators, sensors, physical entities, and network links.
$r(n)$	Amount of a resource r for the node n.
$r(l)$	Amount of a resource r for the network link l.
$c(t)$	Consumption of a resource by the task t for its execution. For each resource consumption c, there exists a corresponding resource r.
$l[x, n]$	Indicates that a node n can communicate with a sensor or an actuator x through a network l.
$t[s]$, $t[a]$	Indicates that a task t consumes data from a sensor s or controls an actuator a.

TABLE I
CONSTANTS OF THE OPTIMIZATION PROBLEM.

$n[t]$	Indicates that a task t is allocated to a node n.
$p[n]$	Indicates that a node n is attached to a physical entity p.

TABLE II
VARIABLES OF THE OPTIMIZATION PROBLEM.

nutrient controller, as well as to other physical entities of the hydroponics room.

C. Optimization Metrics

Once the physical and cyber entities, as well as data flows, are defined, the modeler needs to express what must be optimized and what are the constraints to satisfy with regards to the computation of node placement and task allocation. Having different use cases or different domains implies having different metrics to optimize, such as, e.g.: the network latency, the energy consumption, the data transmission cost if third-party networks are involved, the availability of the network, or the maintenance cost.

In the ontology, the modeler can create or reuse metrics classes and connect them to tasks using the object properties *maximizes* and *minimizes*. Figure 7 provides an example where the `FilterVerifier`, which is a task of the tank controller physical entity defined in Figure 4, must have the minimum `Latency` (i.e., the minimum time for receiving data from sensors and sending orders to actuators) and the maximum `Availability` (i.e., the maximum uptime). These metrics will later be optimized based on the data properties of the sensors, the actuators and the network links. Still in the example shown in Figure 7, `Availability` will be based on the availability of the networks and sensors; and `Latency` will be based on the latency of networks.

Similarly, the modeler can express constraint on metrics, such as the minimum availability of a task, by attaching constraint concepts to tasks. Data properties defined in the XML Schema Datatypes[9] (e.g., `minInclusive` or `maxInclusive`, that represents a minimum or a maximum value) can then be used to parameterize the constraint by setting the range or the accepted values for the constraint. This is for instance illustrated in Figure 7 with the specification that the `SpillVerifier` task should be available 95% of the time.

[9]https://www.w3.org/TR/xmlschema11-2

In addition of setting the metrics to optimize for each task, the modeler is provided with the ability to define metrics to optimize globally for the whole network.

V. OPTIMIZATION

Given all the elements defined in the ontology, we have to build a constrained optimization problem for the computation of the node placement and task allocation. In a nutshell, the *node placement* consists in finding how many computational nodes of which type (e.g., Raspberry PI) must be attached to which physical entities, and the *task allocation* consists in finding onto which of these nodes each task is going to be deployed. As detailed below, the computation of the solution relies on: (i) the conversion of the ontology-based formalization into an abstract optimization problem –i.e., a formulation of a constraint programming problem– (§ V-A), followed by (ii) the translation of this abstract representation into code for a given optimizer implementation, such as Gecode[10] or Choco[11] (§ V-B).

A. Building the Optimization Problem

To build the optimization problem, the ontology concepts are mapped into constraint programming concepts. Roughly speaking, a formal constraint programming problem is composed of a set of constants, a set of variables, a set of objectives to minimize or maximize, and a set of constraints [39]. They all derive from the ontology definition as follows.

a) Constant and variables: The problem inputs, i.e., the concepts defined in the ontology for representing the physical entities, cyber entities and data flows, are converted into the constants defined in Table I. The problem outputs, i.e., the allocation of tasks to nodes and the association of nodes and physical entities are represented by the variables defined in Table II.

[10]http://www.gecode.org
[11]http://www.choco-solver.org

b) Objective functions: As exposed in § IV-C, the modeler defines the metrics to optimize for each task, using the maximize and minimize object properties. Generating the set of objective functions is thus straightforward; we just have to browse the ontology to find which concepts need to be optimized for each task.

As described in § IV-C, different use cases imply different metrics and thus different objective functions. The translation of a metric into an objective function must be defined by the modeler at some point, or be part of a set of pre-defined translations common to various use cases (e.g., network latency or energy consumption). For example, if we note $\alpha(l)$ the latency of a network l, minimizing the network latency for a task t will be expressed as minimizing the sum of the latencies of all network links used by the node n to communicate to all sensors and actuators involved in the data flow of task t, if t is deployed on node n:

$$\min \sum_{n,l \in \mathcal{N},\mathcal{L}} n[t] \times \alpha(l) \left(\sum_{s \in \mathcal{S}} t[s] \times l[s,n] + \sum_{a \in \mathcal{A}} t[a] \times l[a,n] \right)$$

Similarly, maximizing the availability of a task t will be expressed as follows, with $\beta(l)$ and $\beta(s)$ being the failure rates of a network l and a sensor s, respectively:

$$\max \sum_{n,l,s \in \mathcal{N},\mathcal{L},\mathcal{S}} t[s] \times n[t] \times l[s,n] \times \beta(l) \times \beta(s)$$

Regarding global objectives, the principle is similar: instead of minimizing/maximizing an objective function for one task, we minimize/maximize the sum of an objective function f over the set of tasks $\mathcal{T}$: $\sum_{t \in \mathcal{T}} f(t)$.

c) Constraints: In the ontology, some constraints are explicitly defined by the modeler (e.g., the minimum failure rate for a task) and some constraints are added *ex-nihilo* for consistency reasons, based on the ontology data properties (e.g., the resource consumption of the tasks on a node can not exceed the resource of the node).

Most of the task constraints in our problem are conditional, i.e., if a task is deployed on a node, we want to ensure that the task needs are satisfied. This can be expressed using *reified constraints* [40], i.e., the composition of a constraint and a binary variable that denotes its true value. For example, the following constraint expresses that if a task t is mapped on a node n, we have to make sure that each sensor s feeding the task t has a data production rate $\gamma(s)$ that satisfies the minimum data rate constraint $\gamma(t)$ expressed on the task t:

$$\forall s,t \in \mathcal{S},\mathcal{T}, n[t] \Rightarrow \gamma(s) \geq \gamma(t)$$

Similar constraints are defined for all the pre-defined metrics provided by the ontology, e.g., failure rate β or network latency α:

$$\forall s,t \in \mathcal{S},\mathcal{T}, n[t] \Rightarrow \sum_{l \in \mathcal{L}} \beta(s) \times \beta(l) \times l[s,n] \geq \beta(t)$$

$$\forall n \in \mathcal{N}, n[t] \Rightarrow l[n,s] \times \alpha(l) \times t[s] \geq \alpha(t)$$

Regarding the *ex nihilo* constraints, the following rules ensure that the overall resource consumption never exceed the available resources:

$\forall n \in \mathcal{N}$, each node capacity is not exceeded:

$$\sum_{t \in \mathcal{T}} n[t] \times c(t) \leq r(n)$$

$\forall l \in \mathcal{L}$, each network link capacity is not exceeded:

$$\sum_{n,s \in \mathcal{N},\mathcal{S}} l[n,s] \times \sum_{t \in \mathcal{T}} t[s] \times n[t] \times \gamma(t) + \sum_{n,a \in \mathcal{N},\mathcal{A}} l[n,a] \times \sum_{t \in \mathcal{T}} t[a] \times n[t] \times \gamma(t) \leq \gamma(l)$$

d) Node placement and task allocation conflicts: In the stated optimization problem, we defined many variables based on the set of nodes $\mathcal{N}$. However, in our problem, the set of nodes is the result of the optimization process and is thus not known *a priori*. Unfortunately, the two sub-problems are not separable: the allocation of a task to a node depends on the node location while the location of a node depends on the tasks that run on it. As a consequence, it is not possible to compute the set of nodes before performing the task allocation. To ensure that our problem can still be modeled as a constraint problem, we change the problem by creating a virtual set of nodes defined as follows:

- Each physical entity $p \in \mathcal{P}$ has one node of each type (i.e., all physical entities provide all types of node);
- For encouraging the co-location of tasks, a new objective function is added to the multi-objective problem for maximizing *node utilization*, i.e., the amount of resource r consumed on each node:

$$\max_{\forall n \in \mathcal{N}} \sum_{t \in \mathcal{T}} n[t] \times \frac{r(n)}{c(t)};$$

- Once a solution is found, the empty nodes are discarded.

B. Building the Constraint Solver Code

In the IoT context, many different optimization techniques, such as linear programming [41], heuristics [42] or meta-heuristics [43], have been proven useful for solving various types of problems. In practice, there exist various tools and frameworks that implement these techniques for modeling and solving an optimization problem. Building an executable code for the stated optimization problem thus consists into translating the abstract optimization formulation into a concrete optimization specification for the target tools. The generated code will be indeed specific to the set of tools targeted for expressing and running the optimization process.

In our case, we have seen that our optimization problem is multi-objective –e.g., when we need to maximize both the availability of one task and the occupancy of nodes–, and non-linear because of constraint reifications. It is possible to convert a multi-objective problem into a single-objective one using *scalarizers*, i.e., techniques for combining mutiple

Concept	Mapped to
Constants	Java primitive types
Variables	BoolVar (e.g., $n[t]$), IntVar, etc.
Arithmetic expressions	ArExpression#sum, ArExpression#mul, etc.
Constraints	Model#arithm, Model#sum, etc.
Reified constraints	Model#ifThen
Single-objective	Model#setObjective
Multi-objective	ParetoOptimizer or scalarization

TABLE III
MAPPING TO CHOCO CONCEPTS

objective functions into a single one [44]. It is also possible to linearize reified constraints by introducing intermediary variables although this is quite inefficient in practice [41]. Our approach for code generation thus targets constraint programming solvers instead of linear programming ones. For our proof of concept, we considered the Choco solver and defined how to map the optimization problem into Choco concepts, as shown in Table III.

In practice, any code generator can be implemented. The formalization of the problem and its actual resolution being well-separated, the custom code generator can be implemented to target any optimization problem solver. For example, a code generator can enforce scalarization or perform automated linearization in the generated code to target a single-objective optimizer or a linear programming solver. If empirical knowledge exists for the considered domain, heuristic-based code generator can be implemented as well.

C. The Modeler as a Feedback Source

In our approach, we want the modeler to be integrated deeply in the optimization process. Domain experts usually have empirical experience of what a good solution may look like, and enabling them to express their intuitions can be a key for adoptions of such automated methods.

- **Feedback on objectives**: the modeler should be able to express what would be good values for some objective functions (e.g., energy consumption of task t should be close to 95%). This technique is known in the literature as a *reference point* [44] and consists in replacing the objective function by the minimization of the distance to the reference point. This technique can also be used as a scalarizer, if the modeler specifies a reference point for all the objective functions.
- **Feedback on variables values**: the modeler should be able to express what would be good values for some variables (e.g., define a good placement for some nodes and some tasks), by converting these variables into constants.
- **On-the-fly refinement**: the modeler should be able to influence the exploration method of the constraint solver. The first solution that satisfies the constraints, or the best solution found after a given computation time, is presented to the modeler for evaluation. The modeler can then reorient the solution by changing the reference points or validate the good parts of the solution by converting $n[t]$ variables into constants.

VI. CONCLUSION

The IoT is called to bring digital smartness to most vertical sectors. Although there already exist many successful deployments of the IoT in the public (e.g., cf. smart cities) and private (e.g., cf. industry 4.0) domains, the configuration of IoT systems remains a challenge. This is especially true when one considers the deployment of the IoT system at scale. Our vision is that distributed systems research and development has a significant role to play toward the democratization of the IoT, which is well illustrated by prior vision research papers for the field (see [45]–[47]).

In our work, we tackle more specifically the complexity of optimizing the configuration of IoT systems at the edge, which is essential to ensure relevant properties like: reduced resource consumption, enforcing privacy, reduced financial costs, *etc*. As a starting point, we build upon ontological engineering that provides us with the formal description of the real world –spanning the physical and cyber entities. The added value of ontologies for the IoT has been acknowledged for long and has shown to enable overcoming the high heterogeneity of the IoT systems. However, our vision is that we need a re-think of how ontologies are exploited in the design of distributed IoT systems. Specifically, we consider that it may serve taming the complexity of the system regarding both the deployment of the cyber entities to monitor and control the physical world, and implementing the relevant distributed computation at the edge. In a nutshell, we consider that provided the ontologies associated with the relevant vertical and digital domains, we may provide domain experts with an overall framework –The LATTICE framework– assisting the development of distributed IoT systems. As a first evidence, this paper has shown how to automate the optimization of the IoT system at the edge provided the abstract specification of the system regarding its constituent physical and cyber entities.

We are currently finalizing an early prototype implementation of the LATTICE framework, which we intend to experiment using the concrete hydroponics use case that we presented in the paper. Obviously, many challenges remain. One is to make the framework sufficiently generic for the large diversity of IoT systems. That is, we need to allow leveraging the most relevant optimization technique for the given IoT system. Another challenge is to make the framework tractable for the domain expert, which we intend to address by working closely with experts of various domains.

REFERENCES

[1] M. N. Kamel Boulos and N. M. Al-Shorbaji, "On the internet of things, smart cities and the WHO healthy cities," *International Journal of Health Geographics*, vol. 13, no. 10, 2014.

[2] E. Jovanov, A. Milenkovic, C. Otto, and P. C. de Groen, "A wireless body area network of intelligent motion sensors for computer assisted physical rehabilitation," *Journal of NeuroEngineering and Rehabilitation*, 2005.

[3] M. Ye, C. Li, G. Chen, and J. Wu, "Eecs: an energy efficient clustering scheme in wireless sensor networks," in *24th IEEE International Performance, Computing, and Communications Conference*, 2005.

[4] K. S. Low, W. N. Win, and M. J. Er, "Wireless sensor networks for industrial environments," in *International Conference on Computational Intelligence for Modelling, Control and Automation*, 2005.

[5] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of things (IoT): A vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X13000241

[6] L. Jiang, L. D. Xu, H. Cai, Z. Jiang, F. Bu, and B. Xu, "An IoT-oriented data storage framework in cloud computing platform," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 2, 2014.

[7] A. Botta, W. de Donato, V. Persico, and A. Pescap, "Integration of cloud computing and internet of things: A survey," *Future Generation Computer Systems*, vol. 56, 2016.

[8] B. D. Ziebart, D. Roth, R. H. Campbell, and A. K. Dey, "Learning automation policies for pervasive computing environments," in *Second International Conference on Autonomic Computing*, 2005.

[9] N. D. Lane, S. Bhattacharya, P. Georgiev, C. Forlivesi, and F. Kawsar, "An early resource characterization of deep learning on wearables, smartphones and internet-of-things devices," in *Proceedings of the International Workshop on Internet of Things Towards Applications*, 2015.

[10] M. S. Mahdavinejad, M. Rezvan, M. Barekatain, P. Adibi, P. Barnaghi, and A. P. Sheth, "Machine learning for internet of things data analysis: a survey," *Digital Communications and Networks*, vol. 4, no. 3, 2018.

[11] R. Ventura, V. Mallet, and V. Issarny, "Assimilation of mobile phone measurements for noise mapping of a neighborhood," *Journal of the Acoustical Society of America*, vol. 144, no. 3, 2018.

[12] K. Pohl, *Requirements Engineering: Fundamentals, Principles, and Techniques*. Springer Publishing Company, Incorporated, 2010.

[13] A. van Deursen, P. Klint, and J. Visser, "Domain-specific languages: An annotated bibliography," *SIGPLAN Not.*, vol. 35, no. 6, Jun. 2000.

[14] V. Devedzić, "Understanding ontological engineering," *Commun. ACM*, vol. 45, no. 4, Apr. 2002.

[15] A. Gyrard, C. Bonnet, and K. Boudaoud, "Enrich machine-to-machine data with semantic web technologies for cross-domain applications," in *IEEE World Forum on Internet of Things (WF-IoT)*, 2014.

[16] A. Gyrard, M. Serrano, and G. A. Atemezing, "Semantic web methodologies, best practices and ontology engineering applied to internet of things," in *IEEE World Forum on Internet of Things (WF-IoT)*, 2015.

[17] T. Teixeira, S. Hachem, V. Issarny, and N. Georgantas, "Service Oriented Middleware for the Internet of Things: A Perspective," in *ServiceWave - The European conference on Towards a service-based internet*, 2011. [Online]. Available: https://hal.inria.fr/inria-00632794

[18] G. Bajaj, R. Agarwal, P. Singh, N. Georgantas, and V. Issarny, "4W1H in IoT semantics," *IEEE Access*, vol. 6, Oct. 2018. [Online]. Available: https://hal.inria.fr/hal-01898506

[19] D. Fensel, F. van Harmelen, I. Horrocks, D. L. McGuinness, and P. F. Patel-Schneider, "Oil: an ontology infrastructure for the semantic web," *IEEE Intelligent Systems*, vol. 16, no. 2, 2001.

[20] M. Missikoff, R. Navigli, and P. Velardi, "The usable ontology: An environment for building and assessing a domain ontology," in *The Semantic Web — ISWC 2002*, 2002.

[21] G. Blair, A. Bennaceur, N. Georgantas, P. Grace, V. Issarny, V. Nundloll, and M. Paolucci, "The Role of Ontologies in Emergent Middleware: Supporting Interoperability in Complex Distributed Systems," in *12th International Middleware Conference (MIDDLEWARE)*, vol. LNCS-7049, Dec. 2011.

[22] A. Bennaceur and V. Issarny, "Automated Synthesis of Mediators to Support Component Interoperability," *IEEE Transactions on Software Engineering*, 2015. [Online]. Available: https://hal.inria.fr/hal-01076176

[23] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the First Edition of the Workshop on Mobile Cloud Computing*, 2012.

[24] M. Chiang and T. Zhang, "Fog and IoT: An overview of research opportunities," *IEEE Internet of Things Journal*, vol. 3, no. 6, 2016.

[25] S. Pradhan, A. Dubey, S. Khare, S. Nannapaneni, A. S. Gokhale, S. Mahadevan, D. C. Schmidt, and M. Lehofer, "CHARIOT: goal-driven orchestration middleware for resilient IoT systems," *TCPS*, vol. 2, no. 3, 2018. [Online]. Available: https://doi.org/10.1145/3134844

[26] P. Varshney and Y. Simmhan, "Demystifying fog computing: Characterizing architectures, applications and abstractions," *CoRR*, vol. abs/1702.06331, 2017. [Online]. Available: http://arxiv.org/abs/1702.06331

[27] OFC, "The 8 pillars of the OpenFog reference architecture," OpenFog Consortium, Tech. Rep., 2017, white paper.

[28] IIC, "Introduction to edge computing in IIoT," Industrial Internet Consortium, Tech. Rep., 2018, white paper.

[29] B. Billet and V. Issarny, "Dioptase: a distributed data streaming middleware for the future web of things," *Journal of Internet Services and Applications*, vol. 5, no. 1, 2014. [Online]. Available: https://hal.inria.fr/hal-01081738

[30] N. K. Giang, M. Blackstock, R. Lea, and V. C. M. Leung, "Developing IoT applications in the fog: A distributed dataflow approach," in *5th International Conference on the Internet of Things*, 2015.

[31] T. B.-L. Y. S. S. C.-S. Huang Z., Lin K.-J., "Building edge intelligence for online activity recognition in service-oriented IoT systems," *Future Generation Computer Systems*, vol. 87, 2018.

[32] O. Skarlat, M. Nardelli, S. Schulte, M. Borkowski, and P. Leitner, "Optimized IoT service placement in the fog," *Service Oriented Computing and Applications*, vol. 11, no. 4, Dec 2017.

[33] R. Fan, J. Cui, S. Jin, K. Yang, and J. An, "Optimal node placement and resource allocation for UAV relaying network," *IEEE Communications Letters*, vol. 22, no. 4, Apr 2018.

[34] Z. Zhao, G. Min, W. Gao, Y. Wu, H. Duan, and Q. Ni, "Deploying edge computing nodes for large-scale IoT: A diversity aware approach," *IEEE Internet of Things Journal*, vol. 5, no. 5, 2018. [Online]. Available: https://doi.org/10.1109/JIOT.2018.2823498

[35] J. J. Benton Jones, *Hydroponics: A Practical Guide for the Soilless Grower*. CRC Press, 1997.

[36] M. I. Alipio, A. E. M. D. Cruz, J. D. A. Doria, and R. M. S. Fruto, "A smart hydroponics farming system using exact inference in bayesian network," in *2017 IEEE 6th Global Conference on Consumer Electronics (GCCE)*, 2017.

[37] D. Saraswathi, P. Manibharathy, R. Gokulnath, E. Sureshkumar, and K. Karthikeyan, "Automation of hydroponics green house farming using IoT," in *2018 IEEE International Conference on System, Computation, Automation and Networking (ICSCA)*, 2018.

[38] K. Janowicz, A. Haller, S. J. Cox, D. Le Phuoc, and M. Lefrançois, "Sosa: A lightweight ontology for sensors, observations, samples, and actuators," *Journal of Web Semantics*, 2018.

[39] F. Rossi, P. v. Beek, and T. Walsh, *Handbook of Constraint Programming (Foundations of Artificial Intelligence)*. Elsevier, 2006.

[40] N. Beldiceanu, M. Carlsson, P. Flener, and J. Pearson, "On the reification of global constraints," *Constraints*, vol. 18, no. 1, 2013.

[41] A. Pathak and V. Prasanna, "Energy-efficient task mapping for data-driven sensor network macroprogramming," *IEEE Transactions on Computers*, vol. 59, 2010.

[42] B. Billet and V. Issarny, "From task graphs to concrete actions: a new task mapping algorithm for the future internet of things," in *MASS-11th IEEE International Conference on Mobile Ad hoc and Sensor Systems*, 2014.

[43] A. Hauswirth, S. Bolognani, G. Hug, and F. Drfler, "Projected gradient descent on riemannian manifolds with applications to online power system optimization," in *54th Annual Allerton Conference on Communication, Control, and Computing*, 2016.

[44] K. Klamroth and T. Jørgen, "Constrained optimization using multiple objective programming," *Journal of Global Optimization*, vol. 37, no. 3, 2007.

[45] J. Cao, L. Xu, R. Abdallah, and W. Shi, "Edgeos: A home operating system for internet of everything," in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, 2017.

[46] E. Damiani, G. Gianini, M. Ceci, and D. Malerba, "Toward IoT-friendly learning models," in *IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, 2018.

[47] K. Iwanicki, "A distributed systems perspective on industrial IoT," in *IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, 2018.