Ontology-Driven Edge Computing*

Konstantin Ryabinin[1][0000−0002−8353−7641] and
Svetlana Chuprina[1][0000−0002−2103−3771]

Perm State University, Bukireva Str. 15, 614990, Perm, Russia
kostya.ryabinin@gmail.com, chuprinas@inbox.ru

Abstract. The paper is devoted to new aspects of ontology-based approach to control the behavior of Edge Computing devices. Despite the ontology-driven solutions are widely used to develop adaptive mechanisms to the specifics of the Internet of Things (IoT) and ubiquitous computing ecosystems, the problem of creating withal full-fledged, easy to handle and efficient ontology-driven Edge Computing still remains unsolved. We propose the new approach to utilize ontology reasoning mechanism right on the extreme resource-constrained Edge devices, not in the Fog or Cloud. Thanks to this, on-the-fly modifying of device functions, as well as ad-hoc monitoring of intermediate data processed by the device and interoperability within the IoT are enabled and become more intelligent. Moreover, the smart leverage of on-demand automated transformation of Machine-to-Machine to Human-Centric IoT becomes possible. We demonstrate the practical usefulness of our solution by the implementation of ontology-driven Smart Home edge device that helps locating the lost things.

Keywords: Ontology Engineering · Internet of Things · Edge Computing · Ubiquitous Computing · Human-Machine Interaction · Human-Centric IoT · Firmware Generation.

1 Introduction

The development and deployment of a ubiquitous computing environment faces many challenges related to the communicability, reconfigurability and context awareness of individual data processing nodes within this environment. There are a lot of ways to tackle these challenges by introducing intermediate layers in the sensor and actuator network, where certain groups of devices are controlled by hubs. Hubs are normally quite powerful nodes capable of complex algorithms execution including different steering and data aggregation techniques based on Machine Learning or Semantic Web. Such nodes are often denoted as "Fog Nodes", and the corresponding set of computation approaches is called "Fog Computing" [11].

* The reported study is supported by Ministry of Science and Higher Education of the Russian Federation, State Assignment No. FSNF-2020-0017 (Research Project of Perm State University, 2020–2022).

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50436-6_23

The responsibility of Fog Nodes includes tracking of the connected sensors and actuators, performing data preprocessing and aggregation, maintaining the temporal data storage and data transfer, as well as providing needed machine-to-machine (M2M) and human-machine interfaces (HMIs). To make the entire network more flexible, hubs should be aware of their usage context [21], which enables smart control over the potentially transient end-point devices. This, however, restricts the use-cases to the star-like network topologies, where hubs become mandatory links inside the data processing chains and can run into potential bottlenecks both in stability and performance.

One of the ways to improve the stability and performance within ubiquitous computing environment is based on the "Edge Computing" [5] concept, whereby the end-point devices (sensors and actuators themselves, so-called "Edge Nodes") leverage their own computing capabilities to partially offload the hubs. But traditionally the Edge Nodes, being an essential part of data processing chains, are unaware of their role in the entire system, and provide neither configuration nor monitoring interface past the hub. As a result, the flexibility, transparency and reconfigurability of each particular sensor subnet (up to the nearest hub) is still an issue: every change or access for monitoring in the sensor network involves the nearest Fog Node, that still has to enumerate and track all the Edge Nodes.

We propose to push the intelligent capabilities to the edge of the sensor / actuator network making the Edge Nodes as smart, as possible. For this, we utilize ontology engineering methods and means, which allow us to generate the firmware for the corresponding devices making their functioning be driven by the reusable formal knowledge representation model. Previously we successfully applied ontology engineering methods to automate firmware and middleware generation for the devices within the ecosystem of the Internet of Things (IoT) [14]. The current work is devoted to Edge Computing, entirely governed by ontologies. In this case, changes in the firmware of edge devices are completely avoided in the favor of changing the functioning model stored as an ontology.

Compared to the traditional firmware-based approach, ontology-driven Edge Computing ensures the semantic protocols, which allow on-the-fly reconfiguration of edge devices, inspecting their roles and functioning patterns at runtime (by means of self-documentation capabilities), transparently monitoring all the stages of their work (including the monitoring of the intermediate data and their transformations, should it be required). Moreover, if required, ontologies enable the flexible interconnection within the sensor network and the autonomous functioning of intelligent Edge nodes, when no network facilities are available. All this in turn, appears to be a powerful advantage to build IoT ecosystem upon, since the ontology-driven access allows to establish ad-hoc human-machine interaction sessions with any IoT device on demand. Thereby the human-it-the-loop scenario is supported as a key for Human-Centric IoT [2].

Our goal was to make a step towards hardware implementation of task ontologies by organizing the ontology-driven functioning of the light Edge devices based on very resource-constrained microcontroller units (MCUs) like ESP8266

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50436-6_23

(80 KiB RAM, 80 MHz CPU), ATmega328 (2 KiB RAM, 16 MHz CPU) or even ATtiny45 (256 B RAM, 8 MHz CPU).

We implemented the suggested approach within SciVi Smart system [14] (`https://scivi.tools`) and tested it by developing a practically useful IoT device.

2 Key Contributions

In this paper, we present an approach to organize Edge Computing governed by ontologies. The main idea is to replace the traditional firmware of edge device by the ontology reasoner, while the underlying extensible ontology describes the schematics, role and functioning principles of this device.

The following key results of the conducted research are presented:

1. The unified ontology-driven approach to perform configurable and inspectable computing on Edge devices within sensor/actuator network in a unified intelligent way. This allows not only to improve the M2M interconnections, but also to automate the transforming M2M IoT into Human-Centric IoT by means of ad-hoc monitoring and steering Edge devices.
2. The ontology "cognitive compression" method, whereby all the redundant information is trimmed, yet the essential structure of ontology remains retrievable and preserves its semantic power. Removing the excessive ontology nodes and relations, using the topological sorting for the remaining ones and applying multilevel structure layout to describe data flow chains in observable and concise form we managed to fit them in the RAM of tiny MCUs.
3. Software to generate extra-lightweight configurable reasoner for ontologies compressed by the method (2). This reasoner is written in C++ (some parts are written manually, but some are generated automatically according to the user's preferences) and can run on a wide range of MCUs, including ESP8266, STM32, ATmega and even ATtiny.
4. Practically useful implementation of the suggested approach by creating ontology-driven Smart Home Edge device that can help to find lost or forgotten things.

3 Related Work

Ontology engineering appears to be a powerful methodology to leverage interoperability of heterogeneous devices within IoT ecosystems [16], as it brings advanced semantics and context-awareness to the ubiquitous computing [10,19]. The emerging convergence of IoT and cognitive technologies is denoted as Semantic Web of Things (SWoT) [7]. Traditionally, only M2M interaction is considered, and ontologies are mainly used to describe the properties and capabilities of the devices to automate their communication and coordination for the sake of cooperation [7,3,16,21]. However, it is nowadays obvious that the evolution of

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50436-6_23

IoT is not limited by M2M, but also requires technologies to facilitate human-in-the-loop scenario. This direction of IoT is called Human-Centric one [2] and demands both semantic technologies and advanced software and hardware HMIs, including tangible ones [6].

Ubiquitous computing environment traditionally includes 3 main levels of computing: Cloud [11], Fog [11] and Edge [5]. All the levels are characterized by the distributed data processing, but the computation power and energy consumption decrease dramatically form the Cloud to the Edge, so each level requires its own approaches both in hardware and in software. In this paper, we focus on the Edge level.

As stated by K. Sahlmann et al., the key feature required by Edge devices is their adaptiveness that can be reached by using ontology-driven solutions [17]. But the main obstacles to the implementation of such solutions are storage capacity and computing power of the Edge nodes [9,16,18,12,20]. The traditional ontology reasoners are well performed on the desktop computers and capable to run on smartphones [8,4], but still cannot even be started on MCUs [18] despite rapid evolution of Edge hardware. Main problems of Edge devices, which hinder straightforward use of ontology-driven techniques, are the following [9,16,18,12,20]:

1. Low RAM capacity. It often appears to be impossible to arrange all the needed data structures in the Edge device memory, not even the ontology itself.
2. Low CPU frequency. Even if the ontology and related supplementary data fit in the device memory, reasoning process takes then too much time and appears to be practically useless as it is incompatible with the real-time tasks (while Edge Computing often requires real-time operation).
3. Low power. As the reasoning requires a lot of computations and has considerable memory footprint, it is an energy consuming process. But Edge devices are often autonomous, so should consume as less power as possible, or their power sources will drain too fast.

X. Su et al. tackle these problems by introducing so-called "Entity Notation": the concise format for knowledge representation and network package payload encoding [20]. According to the evaluations provided, this format enables drastic reduction of the ontology size compared with standard OWL or RDF notation (compression ratio compared to RDF is about 20 times), while the computational burden and energy consumption decrease as well. This format enables efficient interconnection for MCUs and can be encoded/decoded in a straightforward way, but it still cannot ensure the fitting of entire ontology into the device RAM for the full-fledged reasoning.

Another promising way to fit ontologies to the Edge device memory is proposed by K. Sahlmann et al. [16]. But the role of ontologies is limited to describing the capabilities of Edge devices without affecting their behavior. In the later work K. Sahlmann et al. proposed ontology-driven Edge device virtualization [17]. This appears to be a step further towards ontology-driven Edge

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50436-6_23

Computing, however the reasoning process takes place on the hub (Fog) devices, which have much more computational power compared to Edge devices.

H. Dibowski et al. presented a very elaborate work about the usage of ontologies to describe the device capabilities, but also to retrieve devices, select their operation mode, parametrize them, and automatically evaluate their interoperability [3]. However, the reasoning takes place on the Fog node as well (authors used a quite powerful PC as an aggregator for a large amount of IoT devices).

C. Seitz et al. suggest embedding an ontology-based expert system to automate diagnosis of different anomalies within IoT and robotics systems [18]. The software solution proposed by Seitz et al. performs well on the resource constrained embedded devices running under Linux-based operating system. However, the memory consumption of this software reaches several megabytes, so it is suitable for microcomputers (like e. g.Raspberry Pi), and not for MCUs.

Very impressive results were obtained by H. Abdulrab et al., who developed so-called "Ontology Mediators" – the semantic integration components for ubiquitous computing environment [1]. This is a special model-driven middleware to enable transparent interconnection of different devices, including the Edge ones (running under ECOS). The interconnection based on Ontology Mediators comprises device communication as well as fusion and smart transformation of sensors' data. In fact, this work is the closest one to what we want to achieve, since Ontology Mediators are governed by the knowledge model and the target hardware are very resource-constrained. However, this solution foremost addresses the M2M communication, while the human-in-the-loop scenario is not elaborated.

The main distinctive features of our approach are the following:

1. We suggest full-fledged ontology-driven Edge Computing solution, assuming the behavior of Edge Computing devices (e. g. sensing, data processing, actuation and communication) is fully controlled by task ontologies. Thereby we a make a step towards hardware implementation of ontologies.
2. We target not only microcomputers like Raspberry Pi, but also extremely resource-constrained MCUs (e. g. ESP8266 and even ATtiny45).
3. We focus not only on M2M interconnection, but also on human-in-the-loop scenario (utilizing the semantic power of ontologies to leverage the creation of ad-hoc HMI with Edge devices to enable on-demand device monitoring and steering) and on autonomous functioning of Edge devices (when no network and correspondingly no external steering is available).

4 Proposed Solution

4.1 Background

In our previous research we used ontology-driven solutions to automate the calibration and monitoring of IoT sensor-based devices by means of Smart system SciVi scientific visualization tools [15], as well as to create custom hardware

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50436-6_23

HMI based on IoT technologies [14]. SciVi provides the high-level user interface to describe the behavior and interconnection of custom IoT devices and of middleware to glue together different parts of IoT ecosystem in a graphical form using data flow diagrams (DFDs). Moreover, SciVi enables the automatic firmware or middleware generation for IoT devices according to these DFDs.

Data processing operations, visualization features and code generation mechanism of SciVi are governed by the set of underlying ontologies organized as a repository. The detailed description of this process can be found in [13,15,14]. But until now the firmware for IoT devices generated by SciVi was completely imperative: it implements algorithm described by DFD in a traditional sequential fashion.

In the present work we extend this mechanism by the new module that enables generation of embedded reasoners capable of running direct on the IoT Edge devices and turning them into ontology-driven ones.

4.2 Proposed Ontology-Driven Edge Computing Pipeline

The main idea of ontology-driven Edge Computing is schematically presented in the Fig. 1 and described below.

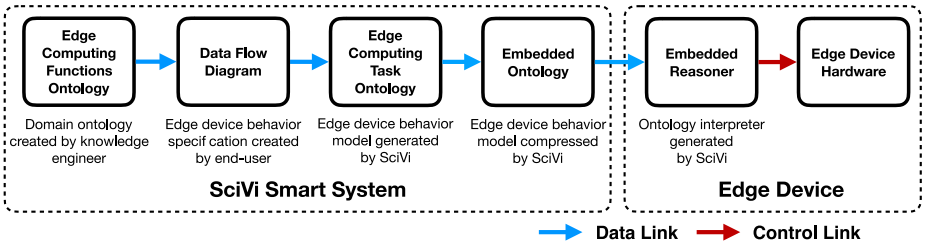

Fig. 1. Ontology-driven Edge Computing (arrows represent data links).

We propose the following lifecycle of the ontology-driven Edge device. First, the embedded reasoner should be generated and installed on MCU. This step is normally performed just once, as a single reasoner covers wide variety of tasks the MCU can be used for. Then, Edge device should be assembled by attaching required sensors and actuators to the MCU. This step can be repeated whenever the device is upgraded. After that, the user should describe the device behavior as DFD within SciVi. This step can be repeated whenever the device role in the computation process should be changed. DFD is automatically transformed into the task ontology that is cognitively compressed and stored in the concise binary format we call EON (Embedded or Edge ONtology). EON-encoded ontology is transferred to Edge device using wired or wireless connection. Afterwards, the reasoner on the device' side starts working, so the device performs described actions. Custom monitoring and steering of the device is allowed in SciVi via special queries to the embedded reasoner. This step is normally repeated multiple times during the device usage, facilitating the human-in-the-loop scenario.

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50436-6_23

To test the proposed approach, we created relatively simple but useful IoT device that is "a reusable pager for the silent things". There are a lot of things (for example, suitcases, glasses, wallets, keychains, etc.) you may want to find in a unified way by a ring signal, just like calling a lost cell phone. The pager we created is removable/reusable; can be attached to the "silent thing" and "called" anytime: it maintains a WiFi access point and buzzes whenever someone connects. The pager also has a "night mode": when the illumination is low, the frequency of buzzer signal decreases, so it is perceived quieter. Thanks to the ontology-based functioning, the base frequency of buzzing can be altered according to the particular thing the pager is attached to, so the user will be able to distinguish, which one is "responding" to the call. We used the following hardware: ESP8266 MCU with a built-in 802.11 (WiFi) capable communication module as a core, passive buzzer HW-508 to play sound and photoresistor VT90N2 to measure illumination strength. These components are very common and cheap, which make it easy to reproduce the results described in the paper.

4.3 Edge Computing Functions Ontology

First of all, we propose an extensible domain ontology of Edge Computing functions (hereafter denoted as D). In fact, this ontology matches the integration of semantic filters ontology and electronic components / middleware ontology created by knowledge engineer within SciVi Smart system described in [14]. The ontology D describes available actions, data processing filters, etc.

Fragment of this ontology applicable for the test case is shown in the Fig. 2. This fragment represents WiFi Access Point being a "Comms" (states for "Communications") element, having the string-typed network SSID and password settings, as well as the output of the numerical amount of connected clients.

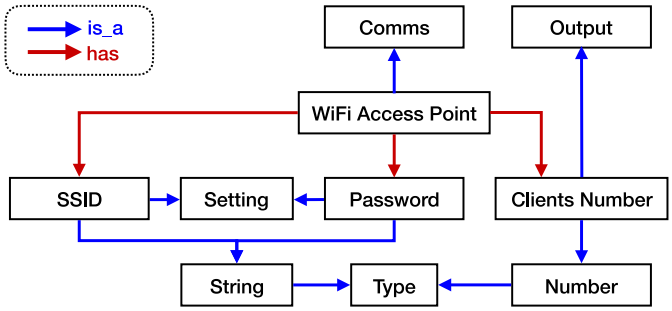

Fig. 2. Fragment of Edge Computing Functions ontology.

4.4 Data Flow Diagram

Based on the ontology D, SciVi automatically generates a set of instruments enabling end-user to compose DFD and thereby visually describe the behavior of Edge device. DFD describing the behavior of the test case Edge device is shown in the Fig. 3.

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50436-6_23

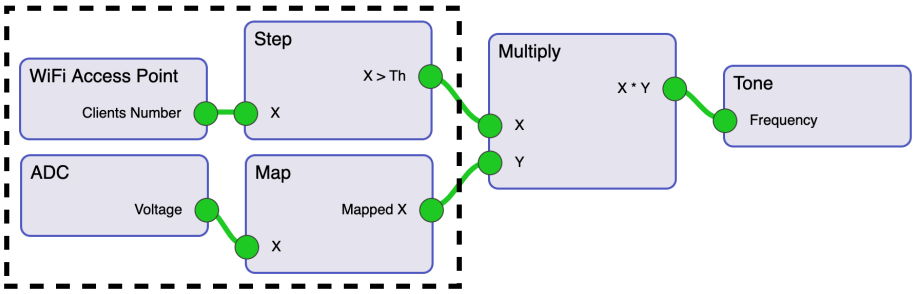

Fig. 3. Data flow diagram representing the behavior of Edge device.

Each node represents the individual data processing stage and links represent the data transfer. It is worth noting, that each node has a corresponding ontology fragment under the hoot, like the one shown in the Fig. 2 (which matches the "Wifi Access Point" node). The diagram defines simple algorithm of Edge device behavior: the number of connected clients, obtained from the Wifi Access Point is tested against the threshold 1 ("Step" node), and multiplied ("Multiply" node) by the value obtained from the ADC (the photoresistor is connected to) and mapped to [100, 1000] ("Map" node). The actual quotients like threshold value and mapping segment borders are tuned in the settings window, that opens by clicking the corresponding node, so they are not depicted in the figure. The multiplication result is then used as a frequency of tone ("Tone" node) played back by the device' buzzer. The part of DFD highlighted by the dotted line will be discussed in the next section.

In real-world, users can define more complex DFDs and, if the provided set of nodes is not enough for them, new ones can be easily added by changing the ontology D instead of modifying the SciVi core source code.

4.5 Edge Computing Task Ontology

The user-defined DFD is than used by SciVi Smart system to automatically create the task ontology (hereafter denoted as T). In fact, this ontology is the representation of DFD in an ontological form.

Each node of T can be computable, this means, it can have an attribute defining a specific function that should be evaluated during the reasoning process to determine the node's value. The nodes can be chained by `use_for` relations, which denote the data flow.

It must be noted, that only the ontology T should be stored and handled on the Edge device side. The ontology D can be stored elsewhere, for example, on the Fog device or even in the Cloud.

Fragment of task ontology T, generated according to the above mentioned DFD and controlling the device is shown in the Fig. 4. This fragment corresponds to the part of DFD highlighted with the dotted line in the Fig. 3. Nodes filled gray and relations depicted by dotted arrows are not stored on Edge device, whereby nodes filled white and yellow, and the relations depicted by solid lines

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50436-6_23

are stored on Edge device. The difference in storing the white and yellow nodes is discussed in the next section.

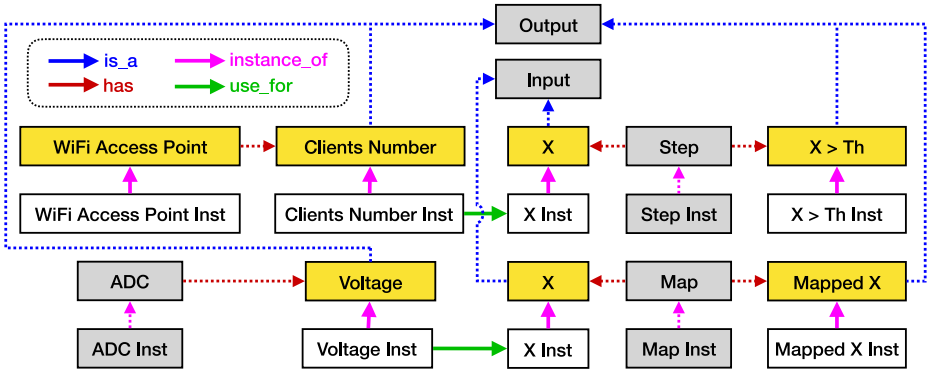

Fig. 4. Fragment of task ontology describing the behavior of Edge device.

4.6 Cognitive Compression and Embedded Ontology

Task ontology T should be uploaded to the target device to govern its functioning. However, it should first be compressed to fit the Edge device memory.

As stated in [18], size of the ontology is the most significant parameter that should be optimized when it comes to reasoning on the Edge. Speaking about MCU, capacity of the memory is often restricted to less than 1 KiB (e. g. popular MCU ATtiny45 possesses 256 B of RAM and 256 B of EEPROM). Traditional formats like OWL or RDF are not even close to be handled on such hardware [20].

Searching through the available information resources, we were unable to find any format that would enable fitting a solid meaningful ontology into less than 256 B. So, we developed our own representation format called EON. It is a concise binary format highly optimized for Edge Computing tasks. The EON format is created special for task ontologies (T) trimming all the redundant information and assuming that, if required, this information can be unambiguously restored with a help of the domain ontology (D). We call this trimming "cognitive compression", and this is an essential part of EON.

Encoding and decoding of ontologies representation in EON format can be expressed in a formal way as follows: $T_{EON} = \sigma(\kappa(T, D)), T = \kappa^{-1}(\sigma^{-1}(T_{EON}), D)$, where σ denotes serialization into EON binary form; κ – cognitive compression; σ^{-1} – deserialization to traditional form e. g. OWL; κ^{-1} – cognitive decompression. T_{EON} denotes task ontology stored in EON format, while T and D are represented in some traditional format e. g. in OWL.

The function σ takes an ontology as a set of nodes, their attributes and relations, and dumps it to a sequence of bytes. This sequence is chunk-based and contains only two chunks (preceded by length in bytes): relations chunk and attributes chunk. It must be noted, that to reduce the size of the byte sequence, we do not store the names of nodes or relations inside it, just integer identifiers

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50436-6_23

(IDs). The names are actually not needed for reasoning on Edge device, so they are trimmed by the function κ.

Relations chunk stores triplets similar to RDF, but represented by integer IDs only. Each triplet is packed into 16 or 24 bit depending on the size of ID. In the current form EON is limited by 64 nodes inside T (after cognitive compression), 65536 nodes inside D and 9 relation types (the possible bitwise layouts are discussed below). These limits appear to be sufficient for computing tasks performed on the low-performance Edge devices. In case of more complex tasks (performed on correspondingly more powerful devices) we can extend EON relation triplets to 32 bit, which will enable handling significantly larger ontologies.

Attributes chunk stores key-value pairs, where key is the integer ID of node and value is its attribute. Normally, attributes are the functions that should be evaluated to calculate the result associated with the node in terms of data flow. Functions may contain operators' ID, nodes' IDs and constants. The table of available operators is tied to the reasoner and is discussed in the upcoming section. If the function contains node ID, the corresponding result of this node is substituted during the reasoning process. Constants can be of different data type. Right now, numerical types (signed and unsigned integers up to 32 bit and 32 bit float) and null-terminated ASCII strings are supported.

Functions are stored in the postfix notation to reduce the entry size. Each function component and each attribute have a distinctive bit marker. Thereby, no length values are stored to reduce the size of entire sequence.

Task ontology T describes the data flow inside Edge device and communication protocols with the other computing nodes in the network. In the data flow, order of operations is crucial. As the operations are encoded to the ontology nodes' attributes, we suggest storing their order implicitly as the order of nodes. As seen in the Fig. 4, we apply topological sorting to the nodes, so the data flow represented by `use_for` relations is aligned from left to right, building a multilevel structure layout of ontology graph representation. This layout is both human- and machine-readable, incorporating the knowledge about the operations order. When dumped to EON format, this partial order of nodes is preserved, so the reasoner can evaluate the nodes one by one, and any time the next node requires the data from the previous ones, these data are already computed. This reduces both the size of result ontology and the computation time.

Next, when preparing T to dumping to EON, we trim all the redundant knowledge that can be unambiguously restored by traversing the domain ontology D and contains no descriptions of actions essential for Edge device functioning. The trimmed nodes are filled gray in the Fig. 4, the trimmed relations are shown as dotted arrows. Currently we remove all relations except `instance_of` (because it denotes the connection of T and D and thereby is needed to restore the trimmed knowledge) and `use_for` (because it directly denotes the data flow). It must be noted, that EON, as an ontology representation format, can support much more relations, letting space for future modifying (e. g. generalizing) the cognitive compression algorithm.

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50436-6_23

The names of nodes and relations are also trimmed (replaced by the IDs), so the cognitive decompression is possible only if the ontology D is presented. However, as long as the cognitive decompression takes place in SciVi that runs on the Fog Node or in Cloud, access to D is not an issue. To reduce the size of resulting byte sequence, we enumerate the nodes filled white in the Fig. 4 starting from 1, but to make the decompression possible, we keep the IDs from the ontology D for the nodes filled yellow. So, for each `instance_of` relation, the source node is the one from T (so its ID is just a number of node in the multilevel structure layout of T), and the destination node is denoted by its ID in D.

To represent the IDs in the EON byte sequence, following layouts are used (`s` denotes bit of source node ID, `r` – bit of relation ID, `d` – bit of destination node ID):

1. `s s s s s s 0 r r r d d d d d d`
   (regular relation for the nodes inside T).
2. `s s s s s s 1 0 d d d d d d d d`
   (`instance_of` relation joining T and D, if D has no more than 256 entities).
3. `s s s s s s 1 1 d d d d d d d d d d d d d d d d`
   (`instance_of` relation joining T and D, if D has from 256 to 65536 entities).

Last, but not least, we optimize the storage of constants by finding the smallest type including particular value (e. g. the value 1.0 is treated as 8 bit integer, while 1.1 – as 32 bit float).

In our test case, the ontology, which fragment is shown in the Fig. 4, took only 80 bytes after cognitive compression and dumping to EON (this appears to be more than 800 times smaller than storing it in OWL format).

4.7 Embedded Reasoner

The aim of embedded reasoner is to ensure the evaluation of data flow represented by the ontology T in the EON format. The architecture of the reasoner is shown in the Fig. 5, where data links depict the transfer of data between the reasoner modules, and control links represent the transfer of commands.

To enable portability across different MCUs, the reasoner is implemented in C++. Currently, we have tested it on ESP8266 (within WeMos D1 mini platform), ATmega328 (within Arduino platform) and ATtiny45 (as a standalone MCU). The compilation process is automated within SciVi Smart system [14].

The **Functions Module** steers the hardware and provides the table of available operators, which enables the MCU's instruction set to be used within ontologies. The code of this module is generated automatically by SciVi according to the user's choice of libraries that should be supported in the particular reasoner. This configurability allows reducing the size of reasoner's binary representation (e. g. ATtiny45 has only 4 KiB flash memory to store the code).

The **Communication Module** allows sending and receiving data and commands to / from SciVi. Basically, 4 commands are supported: `Upload` (to save

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50436-6_23

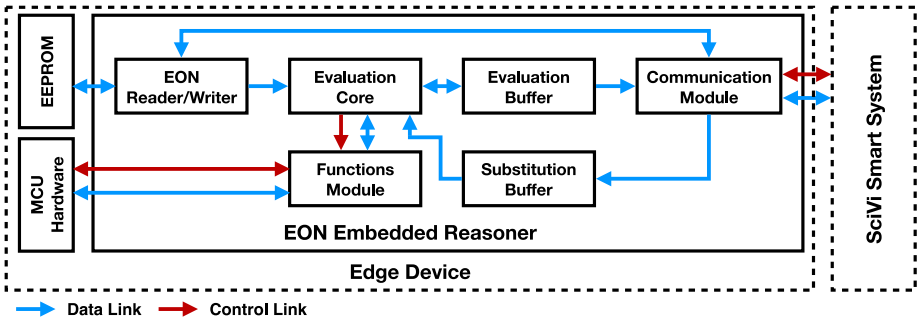

Fig. 5. Architecture of EON embedded reasoner.

new ontology T on the MCU), `Download` (to retrieve T from MCU), `Get` (to receive the particular evaluated result of given node within T) and `Set` (to change the evaluated result of given node by the new one, that was evaluated on the side of SciVi). Communication takes part either via WebSocket over WiFi (in case of ESP8266), via modified set of AT-commands via RS-232 serial port (in case of ATmega328) or via USI SPI (in case of ATtiny45).

The ontology T is stored in the MCU's EEPROM. The reading and writing of EON format is ensured by the **EON Reader/Writer**. The **Evaluation Core** performs traversing the EON-encoded ontology and calls operators provided by **Function Module** whenever needed. Evaluation results are stored in the **Evaluation Buffer**; the results obtained from SciVi (via **Communication Module**) are stored in the **Substitution Buffer**. **Substitution Buffer** allows steering the MCU externally by overriding the results evaluated by the **Evaluation Core**. This in turn enables building ad-hoc user interface with Edge device and opens a gate for the on-demand transformation of M2M functioning into the Human-Centric one.

When the ontology T encoded to EON format and uploaded to Edge device, its embedded reasoner starts executing the tasks described. This is an autonomous process (suitable for M2M functioning scenario), however the end-user can anytime connect to the device using SciVi Smart system to monitor the data flow or even to take control over it. For this, ontology T is retrieved back to SciVi, decoded from EON with help of the ontology D and transformed into the DFD shown to the user. After that, the user can add visualization nodes to this diagram utilizing the visual analytics capabilities of SciVi [15], or change the data links to modify the flow itself. Data for monitoring are retrieved in real-time (using `Get` command discussed above), and the modifications of the data flow take effect immediately (using `Set` command).

The changes in data flow are kept as long as the communication session with Edge device persists. By disconnecting, all the changes are automatically reverted by the embedded reasoner (by wiping the **Substitution Buffer**), so Edge device restores its initial functioning. If the user wants to store the changes, updated DFD should be dumped to task ontology and committed to Edge device.

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50436-6_23

5 Conclusion

In this paper, we demonstrate the new ontology-driven approach to bring more intelligence to Edge devices within the IoT ecosystem. In contrast to the traditional immutable firmware, the ontology-driven approach ensures adaptivity of devices and introduces semantic power to the ubiquitous computing systems. Our approach allows to reduce the developers efforts thanks to the firmware generation. Also it makes the modifications of machine-to-machine and human-machine interactions within IoT ecosystem more clean and simple thanks to the ontology-based task descriptions. We developed the efficient binary format EON and cognitive compression algorithm to represent task ontologies on Edge devices' side, implemented the set of instruments to automate the creation of corresponding ontologies and the lightweight configurable reasoner using the tools of our Smart system SciVi.

We tested our approach by creating a simple yet useful Edge device that helps to locate lost things. SciVi Smart system is an open source project available on `https://github.com/scivi-tools`. The subproject discussed in this paper is available on `https://github.com/scivi-tools/scivi.eon`.

Our approach can be refined by adding the task ontology verification stage to find out whether the particular ontology from the repository is suitable for the device of certain hardware configuration. Also, the embedded reasoner efficiency should be studied in terms of CPU overhead and energy consumption.

We plan to use this approach to create function-reach hardware HMI for complex visual analytics tasks by studying multiparametric modeling of users' communication processes in the Internet social services [14].

References

1. Abdulrab, H., Babkin, E., Kozyrev, O.: Semantically Enriched Integration Framework for Ubiquitous Computing Environment. In: Babkin, E. (ed.) Ubiquitous Computing, chap. 9, pp. 177–196. IntechOpen (2011). https://doi.org/10.5772/15262
2. Calderon, M., Delgadillo, S., Garcia-Macias, A.: A More Human-Centric Internet of Things with Temporal and Spatial Context. Procedia Computer Science **83**, 553–559 (2016). https://doi.org/10.1016/j.procs.2016.04.263
3. Dibowski, H., Kabitzsch, K.: Ontology-Based Device Descriptions and Device Repository for Building Automation Devices. EURASIP Journal on Embedded Systems (2011). https://doi.org/10.1155/2011/623461
4. Guclu, I., Li, Y.F., Pan, J.Z., Kollingbaum, M.J.: Predicting Energy Consumption of Ontology Reasoning over Mobile Devices. Lecture Notes in Computer Science **9981**, 289–304 (2016). https://doi.org/10.1007/978-3-319-46523-4_18
5. Hamilton, E.: What is Edge Computing: The Network Edge Explained (2018), `https://www.cloudwards.net/what-is-edge-computing/`, last accessed 08 Jan 2020
6. Ishii, H.: Tangible Bits: Beyond Pixels. In: Proceedings of the 2nd International Conference on Tangible and Embedded Interaction. pp. XV–XXV (2008). https://doi.org/10.1145/1347390.1347392

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50436-6_23

7. Jara, A.J., Olivieri, A.C., Bocchi, Y., Jung, M., Kastner, W., Skarmeta, A.F.: Semantic Web of Things: An Analysis of the Application Semantics for the IoT Moving towards the IoT Convergence. International Journal of Web and Grid Services **10**(2/3), 244–272 (2014). https://doi.org/10.1504/IJWGS.2014.060260
8. Koopmann, P., Hähnel, M., Turhan, A.Y.: Energy-Efficiency of OWL Reasoners – Frequency Matters. Lecture Notes in Computer Science **10675**, 86–101 (2017). https://doi.org/10.1007/978-3-319-70682-5_6
9. Li, P.: Semantic Reasoning on the Edge of Internet of Things. Ph.D. thesis, University of Oulu (2016), master's thesis, Degree Programme in Computer Science and Engineering
10. Pardo, E., Espes, D., Le-Parc, P.: A Framework for Anomaly Diagnosis in Smart Homes Based on Ontology. Procedia Computer Science **83**, 545–552 (2016). https://doi.org/10.1016/j.procs.2016.04.255
11. Pisani, F., Borin, E.: Fog vs. Cloud Computing: Should I Stay or Should I Go? In: Proceedings of the Workshop on INTelligent Embedded Systems Architectures and Applications. pp. 27–32 (2018). https://doi.org/10.1145/3285017.3285026
12. Ruta, M., Scioscia, F., Sciascio, E.D.: A Mobile Matchmaker for Resource Discovery in the Ubiquitous Semantic Web. In: 2015 IEEE International Conference on Mobile Services. pp. 336–343 (2015). https://doi.org/10.1109/MobServ.2015.76
13. Ryabinin, K., Chuprina, S.: High-Level Toolset for Comprehensive Visual Data Analysis and Model Validation. Procedia Computer Science **108**, 2090–2099 (2017). https://doi.org/10.1016/j.procs.2017.05.050
14. Ryabinin, K., Chuprina, S., Belousov, K.: Ontology-Driven Automation of IoT-Based Human-Machine Interfaces Development. Lecture Notes in Computer Science **11540**, 110–124 (2019). https://doi.org/10.1007/978-3-030-22750-0_9
15. Ryabinin, K., Chuprina, S., Kolesnik, M.: Calibration and Monitoring of IoT Devices by Means of Embedded Scientific Visualization Tools. Lecture Notes in Computer Science **10861**, 655–668 (2018). https://doi.org/10.1007/978-3-319-93701-4_52
16. Sahlmann, K., Scheffler, T., Schnor, B.: Ontology-driven Device Descriptions for IoT Network Management. In: 2018 Global Internet of Things Summit (GIoTS) (2018). https://doi.org/10.1109/GIOTS.2018.8534569
17. Sahlmann, K., Schwotzer, T.: Ontology-Based Virtual IoT Devices for Edge Computing. In: Proceedings of the 8th International Conference on the Internet of Things (2018). https://doi.org/10.1145/3277593.3277597
18. Seitz, C., Schönfelder, R.: Rule-based OWL Reasoning for specific Embedded Devices. Lecture Notes in Computer Science **7032**, 237–252 (2011). https://doi.org/10.1007/978-3-642-25093-4_16
19. de Souza, W.L., do Prado, A.F., Forte, M., Cirilo, C.E.: Content Adaptation in Ubiquitous Computing. In: Babkin, E. (ed.) Ubiquitous Computing, chap. 4, pp. 67–94. IntechOpen (2011). https://doi.org/10.5772/15940
20. Su, X., Riekki, J., Haverinen, J.: Entity Notation: Enabling Knowledge Representations for Resource-Constrained Sensors. Personal and Ubiquitous Computing **16**, 819–834 (2012). https://doi.org/10.1007/s00779-011-0453-6
21. Venkatesh, J., Chan, C., Rosing, T.: An Ontology-Driven Context Engine for the Internet of Things. Tech. Rep. CS2015-1009, Department of Computer Science and Engineering, University of California, San Diego (2015), `https://csetechrep.ucsd.edu/Dienst/UI/2.0/Describe/ncstrl.ucsd_cse/CS2015-1009`, last accessed 08 Jan 2020

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50436-6_23