Extending Network Programmability to the Things Overlay using Distributed Industrial IoT Protocols

Esteban Municio, Steven Latré, *Member, IEEE,*, and Johann M. Marquez-Barja, Senior Member, IEEE

***Abstract*—Current Industrial Internet of Things (IoT) demands are calling for more flexible and programmable networks that ensure high reliability in dynamic mission-critical scenarios. Centralized Software Defined Networking (SDN) offers the high levels of flexibility and programmability that traditional distributed IoT protocols cannot offer. However the use of SDN in IoT is currently not really lifting off due to wireless links unreliability, excessive control overhead and devices' limited resources. In order to reduce the impact of these issues, Whisper enables SDN-like capabilities in IoT by centrally controlling the distributed routing and scheduling planes in the IoT network (Things Overlay). To do so, the Whisper controller sends carefully computed messages compatible with the standardized distributed protocols already running in the network that change the default protocols' behavior. However, as many other SDN-on-IoT approaches, Whisper is currently limited to the IoT network scope and remains as yet another independent network management silo. In this work, we argue that IoT network control should be jointly coordinated by the same SDN instance that also manages the wired segments. In order to do so, we present a new fully programmable solution that shifts the Whisper scope from the edge to the core, deploying and testing such architecture in real-world large-scale testbeds. We use 6TiSCH as Industrial IoT enabler and the ONOS platform to orchestrate all network segments. Finally we report the technical challenges, discussing the lessons learned, and demonstrating the feasibility and suitability of this Whisper-based solution to provide an efficient and programmable end-to-end control over a heterogeneous network domain.**

***Index Terms*—IIoT, SDN, Whisper, 6TiSCH, RPL, Flexibility.**

I. Introduction

The concept of Industry 4.0 heavily relies on the use of Internet of Things (IoT) and Cyber-Physical Systems (CPS) to revolutionize industrial processes [1]. However, besides the requirements of low latency, ultra-high reliability and low power consumption, industrial networks also require to be flexible and programmable in order to cope with and be further tailored to the actual and dynamic industrial automation needs.

This flexibility is already being provided in wired technologies through the use of Software Defined Networking (SDN). SDN enables network programmability and fine-grained resource management that is almost impossible to obtain with traditional distributed network protocols. Through the centralized decisions of a network controller, devices can automatically be configured at runtime to be better adapted to network dynamics. However, industrial wired networks are expensive to deploy (from $100s/ft to $1000s/ft) [2], making them inviable for many potential adopters. Fortunately, the new advances in Industrial IoT (e.g., 6TiSCH, WirelessHART, SmartMeshIP) are allowing the IoT to fulfill the Industry 4.0 demands [3], and thousands of networks are currently being deployed for these purposes [4].

Unfortunately, SDN-like programmability in IoT is still not being commonly used. The highly resource-constrained nature of the IoT devices in terms of energy and link reliability impede an easy and direct mapping of the wired SDN techniques to the IoT world. Although a significant research effort has been done to engineer solutions for these constraints [5], the in-band signaling overhead, the increase in energy consumption and the uncoupling between the routing and scheduling layers remain as open challenges. Among the existing SDN-on-IoT solutions, Whisper [6] "softwarizes" the IoT network (creates the Things Overlay) by altering the standard behavior of the legacy distributed routing and scheduling protocols running already in the network. This is done by artificially injecting protocol messages in the network without the need of adding a new specific SDN-protocol or modifying the firmware in the already deployed nodes.

However, even when IoT networks can become effectively programmable, these solutions are currently limited to isolated IPv6 Low-power Wireless Personal Area Networks (6LoWPANs). In this paper, we argue that Industry 4.0 requires complete end-to-end flexibility that covers both the wired/optical segment and the wireless segment. We aim to answer the question of "*Can Industrial IoT networks be efficiently and jointly sofwarized along with the core network?*" and will discuss why having such holistic control could be interesting for the Industry 4.0 (Section II). In order to do so, we present a new end-to-end Whisper-based solution (Section IV) and report our experiences deploying such solution in real-world large-scale open testbeds (Section V). Finally, we discuss the challenges and suitability of this approach as a new promising Industry 4.0 enabler for end-to-end IoT control over a multi-technology and multi-domain industrial network.

II. Why End-to-End Flexibility?

Industrial automation has been traditionally a rather conservative domain, and higher reliability has often outweighed the flexibility of the wireless links [7]. However current dynamic industrial environments demand new levels of network programmability. For example, energy consumption highly affects network lifetime. Thus, dynamically managing power is an essential element of the new Industrial IoT. It is therefore logical, that networks adapt their own configuration to achieve the

Manuscript received October 29, 2019; revised Dec 6, 2019; accepted February 1, 2020. Date of publication XXX XX, 2020; Paper no. TII-19-4787. (Corresponding author: Esteban Municio.)

Esteban Municio, Steven Latre and Johann M. Marquez-Barja are with IDLab, University of Antwerp - IMEC, Antwerpen 2000, Belgium (e-mail: {esteban.municio,steven.latre,johann.marquez-barja}@uantwerpen.be).

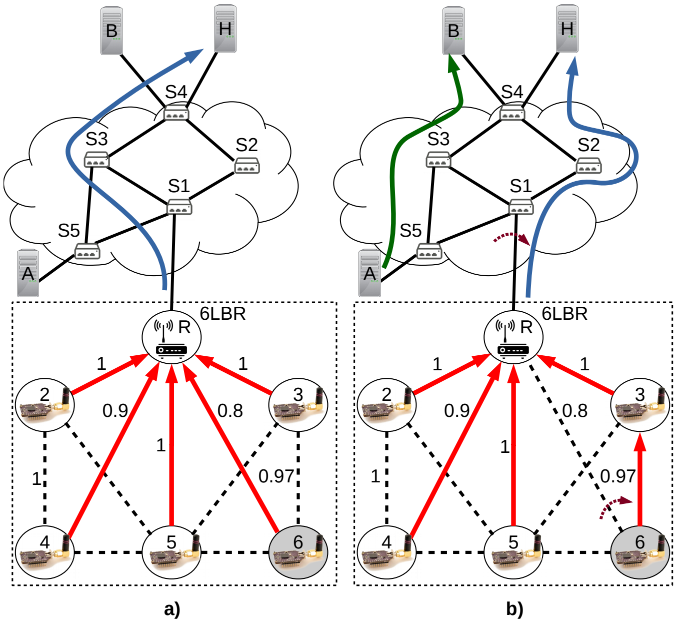


Fig. 1: In Scenario 1 a), the network optimizes energy consumption. Upon network changes b), the flow is reconfigured along the whole network path to maximize reliability.

best trade-off between performance and energy consumption according to the actual needs and scenarios.

In this sense, in order to achieve effective industrial performance, full end-to-end QoS control that includes all wired/wireless network segments is required (e.g., OPC UA and TSN) [8], [9]. This is because Industry 4.0 scenarios imply higher levels of management complexity, where orchestrators have to control different network domains (e.g., Cloud/NFV, fog and edge networks, etc.) and ideally, also the IoT devices [10]. Additionally, the core network is often considered to be always robust and over-dimensioned to cope reliably with large amounts of traffic. But reality is actually far from this, and many issues attributed to unmanageable traffic, failed updates or attacks cause frequent outages and performance drops in wired networks as well [11].

Let us assume Scenario 1 depicted in Figure 1. Imagine that the sensor nodes are monitoring the state of industrial assets in mission-critical or dangerous areas, e.g., through Supervisory Control And Data Acquisition (SCADA). When human presence is not detected, nodes are in power-saving mode and choose the closest neighbor to the 6LoWPAN Border Router (6LBR) as preferred parent to save energy. Ultra-high reliability is not required yet so a Packet Delivery Ratio (PDR) equals to 0.8 is sufficient. This traffic is directed towards the 6LBR with destination Server H in the core network (not necessarily outside the factory) and goes through a level 2 SDN-enabled wired network using any available path (e.g., through S3) as shown in Figure 1 a).

Now suppose that human presence is detected close to sensor 6. In order to have more credible sensor readings from the asset, traffic from sensor 6 requires now high reliability. To achieve this, it seems logical that the data path from node 6 goes now through node 3 (has a higher PDR). However in the wired network, reliability could also be compromised (e.g., a new heavy flow between A and B appears as depicted in Figure 1 b)). Ideally, the controller should react to these events by selecting the path with highest end-to-end reliability according to the new demands (i.e., through node 3 in the 6LoWPAN and through S2 in the wired network).

With this toy example, we argue the importance to have global, flexible control over industrial networks and why we need some kind of joint end-to-end "softwarization" in all network domains. In Section V we will further study this use case, together with other complementary examples.

III. Background and State of the Art

A. How to control Industrial IoT?

Within the Industrial IoT ecosystem, 6TiSCH is currently one of the most growing technologies that combines the Timeslotted Channel Hopping (TSCH) mode of IEEE 802.15.4e with an IPv6-enabled upper stack. This upper stack includes the distributed 6Top Protocol (6P) that manages the scheduling through a Scheduling Function (SF) and the Routing Protocol for Low-power and Lossy networks (RPL), an efficient gradient-based distributed routing protocol designed for resource-constrained devices. Ideally in 6TiSCH, both scheduling and routing layers need to be coupled to provide the required performance to each of the next hops (e.g., capacity and latency in a per hop basis).

However 6TiSCH implements statically defined decisions (i.e., how routing and scheduling planes accomplish a predefined objective function) and the new Industrial IoT requires dynamic on-demand management. Low-power and Lossy Networks (LLNs) are formed by highly resource-constrained devices, where links are unreliable and limited in bandwidth, and the multi-hop wireless mesh topology requires in-band signaling. This implies that bringing SDN into the LLNs is challenging. Although SDN-equivalent centralized approaches have been introduced for 6TiSCH, only concepts and architectures have been provided so far [12]. Outside of 6TiSCH, SDN-on-IoT solutions exist, such as SDN-WISE, IT-SDN and μSDN [13], [14], [15]. However these solutions still rely on a reliable in-band signaling channel and may face important challenges when scaling up the network due to the significant signaling overhead. Additionally, they are based on the non-deterministic content-based ContikiMAC [16], which cannot be strictly defined as an Industrial IoT protocol. These approaches do not cater for the required deterministic performance levels due to the actual uncoupling between their MAC and routing layers.

On the other hand, other hybrid works such as Whisper [6] combine SDN techniques with the distributed IoT protocols present in 6TiSCH. This is the approach we follow in this paper. Whisper offers a trade-off between total control and low overhead. Since it leverages RPL and 6P to exert network control, it is limited to what these protocols can do (e.g., routes are required to form a Direction Oriented Directed Acyclic Graph (DODAG) since RPL is a gradient-based routing protocol). However, control signaling is minimum due to Whisper leverages the messages of the already running

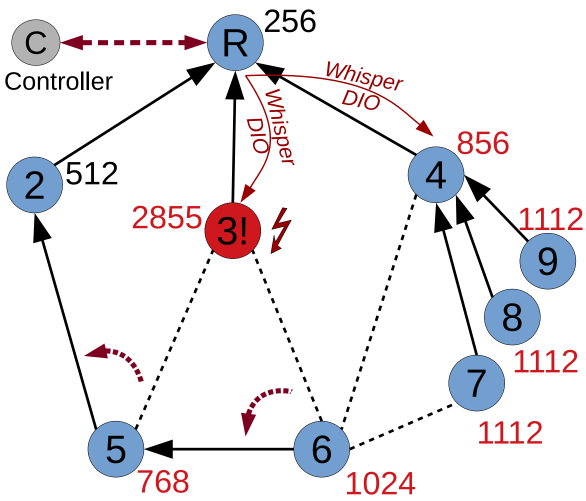


Fig. 2: Whisper alters packet forwarding through the injection of "crafted" RPL messages. Nodes 5 and 6 can avoid to forward packets towards node 3 upon battery depletion, and re-route their traffic through node 2 (to not overload node 4).

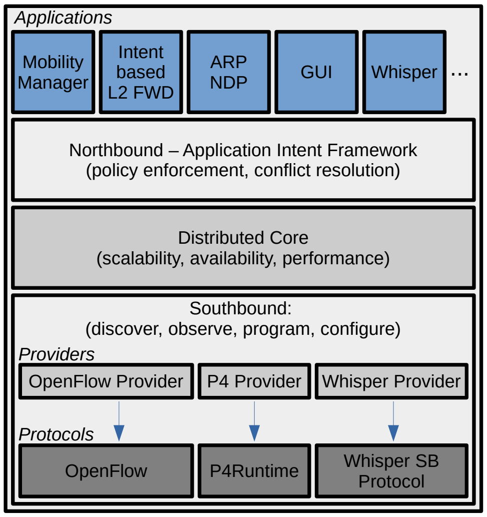


Fig. 3: ONOS architecture.

distributed protocols, and the IoT network can fully perform even without the continuous presence of a SDN controller. This means it is also more energy efficient and scalable that the previous SDN-on-IoT solutions [6].

Figure 2 shows an example of how Whisper works. In RPL each node receives a Rank value which is used to calculate the best path towards the root (R) according to a specific metric, by default Expected Transmission Count (ETX). Since Ranks are distributed through DODAG Information Objects (DIOs), a Whisper controller can inject its own "crafted" DIO messages that alter the standard behavior of the protocol according to the controller's will. Similarly, scheduling is controlled by delivering artificial 6P messages that complement the SF running in the network. By default, 6TiSCH networks use the Minimal Scheduling Function (MSF) [17].

B. How to Orchestrate?

SDN is considered today the tipping point that changed how networks are built and operated. In SDN, traffic can be fine-grained steered according to the centralized decisions of a network controller. This is achieved by allowing the controller to exert direct control over the network devices at runtime through a separated control plane that dictates the required device configuration to fulfill any given policy at any given time. Network Operating Systems (NOS) [18], [19] are already commonly used to program network layers in a platform agnostic manner, decoupling network control from packet forwarding. In order to virtualize the network, NOSs mainly rely on controllers to remotely program the packet forwarding function (e.g., [20]) through a variety of protocols such as OpenFlow [21], NETCONF [22] or P4Runtime [23].

One of the most extended NOS solution in both research and production environments is the open-source Open Network Operating System (ONOS) project [24]. ONOS is a distributed, modular SDN control platform that enables high levels of scalability, availability, and performance in large operator networks [25]. While ONOS is mainly focused on the wired segments, some works have studied how to extend the control to wireless sensor networks as well [26], [27] (e.g., by using the non-industrial IoT SDN-WISE solution).

ONOS independently abstracts network devices from the underlying network architecture in order to permit interoperability between heterogeneous networks. In order to interact with the underlying network, ONOS uses its Providers, which are standalone components that can be dynamically allocated at runtime. Providers translate the device-specific logic to the abstractions used in the upper layers through the device drivers, which implement the device communication protocols (see the complete architecture in Figure 3). The Distributed Core, stores all information maintained by the system (e.g., topology, states, etc.) and provides the upper layers with path computation functions (e.g., to create/compile path Intents).

Finally, the Northbound sublayer manages the network abstractions through flow rules and policies. This sublayer allows applications (e.g., an ARP service, a host mobility manager, etc.) to consume and manipulate aggregated information from the Core sublayer. Application functionality ranges from displaying network topologies (e.g., GUI) to perform complex traffic engineering for different traffic classes (e.g., intent-based forwarding).

IV. Flexibility from the Sensors to the Core

In this section we present the global architecture that enables a joint orchestration of both wired and wireless networks. This architecture integrates in one controller instance different network domains, including the IoT network (e.g., in our case, 6TiSCH). This is done by deploying different Whisper controllers in the 6LBRs to extend the core network virtualization also to the IoT domain. The Whisper controllers communicate with the orchestrator (e.g., in our case, ONOS) to abstract the IoT network. This abstraction allows monitoring and controlling the IoT network in both the routing and scheduling planes. In order to translate high-level control abstractions to actual Whisper primitives, we propose a new Whisper Southbound

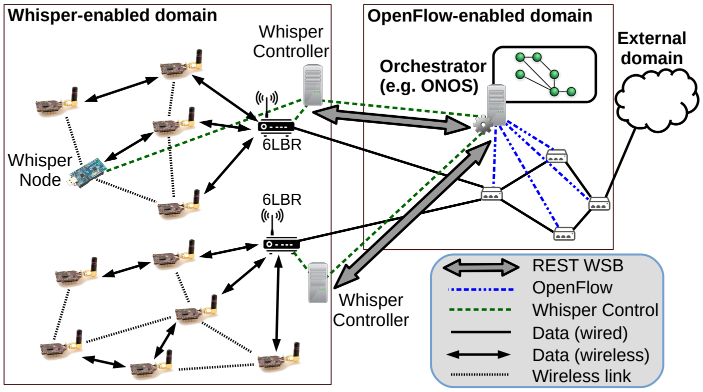


Fig. 4: Orchestration architecture using Whisper extending the SDN control to the Industrial IoT network.

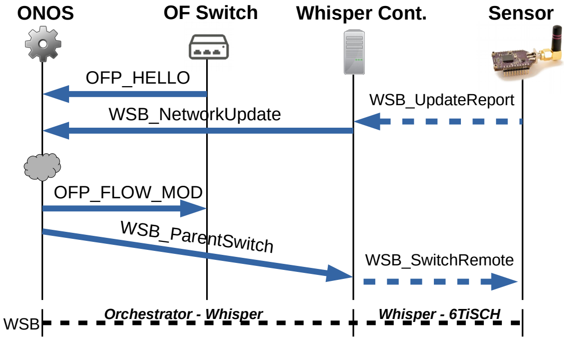


Fig. 5: Example of the messages exchanged, using OpenFlow for the wired segments and the Whisper Southbound protocol (WSB) to control the 6LoWPAN network.

(WSB) protocol (detailed in Section IV-A) which is available through a REST API and which is eventually exerted in the 6TiSCH networks through the Whisper controller located at the 6LBRs. In order to improve network control, the Whisper primitives (e.g., a “crafted” DIO to change a node’s next hop) can be also delivered through Whisper nodes, specific wireless nodes that can be strategically placed in the network to augment the monitor and control capabilities [6] (see Figure 4).

Whisper controllers periodically report to the ONOS controller with network statistics (e.g., topology, link costs, schedules, etc.). This way the ONOS controller updates its internal topology stored in its core, and performs actions according to the policies and applications’ requirements whenever needed. These actions are delivered through OpenFlow to the SDN-capable switches and through WSB to the IoT nodes. Traffic flows are routed through end-to-end Intents from each IoT to its final destination. This way, changes in the traffic paths do not compromise the performance of the flows, since Intents ensure an end-to-end path based on agreed constraints. Upon network changes, the Intent will automatically re-route the flow to accomplish its constraints, both in the wired segment and in the wireless 6TiSCH segment.

A. The Southbound Whisper protocol

The WSB protocol is an enhanced, generalized version of the Whisper primitives described in [6] to make them compatible with a generic SDN controller. Table I describes the most relevant messages[1]. It is divided in two parts:

- The *Orchestrator-Whisper* segment is an abstraction of the full WSB that hides the complexity of the 6TiSCH network to the controller (e.g., Ranks, PDR, etc.). It only consists of 4 messages: ParentSwitch to perform the re-routing of next hop of a sensor node, AddCell/DeleteCell to manage nodes’ schedules and NetworkUpdate, which contains incoming aggregated information from the 6TiSCH network. As they are sent through a bidirectional REST API, arguments are encoded in JSON format with a timestamp and a sequence number, so that missing messages are tracked and retransmitted if necessary.
- In the *Whisper-6TiSCH* segment the protocol is augmented with the characteristics of each specific 6TiSCH network. This means that the Whisper controller translates the abstracted messages from the SDN controller to the actual Whisper primitives needed to perform the SDN controller’s orders. For example, the ParentSwitch message can be translated to one or more messages in the 6TiSCH network (e.g., it could require a SwitchRemote message and a PropagateRank message each of them with specific Rank values, being sent each in a particular order). Likewise, a number of UpdateReports from different Whisper nodes are aggregated at the Whisper controllers in one single NetworkUpdate message destined to the ONOS core. Messages in this segment are sent at byte-level using CoAP [28].

An example of how the architecture operates is depicted in Figure 5. First, ONOS receives information about the state of the network (e.g., links, devices, etc.). Upon decision or when the new requirements are no longer met, the ONOS controller triggers the network changes through OpenFlow in the OpenFlow-enabled switches or through WSB in the Whisper controllers. Upon reception, the Whisper controllers translate the required action to specific 6TiSCH-aware primitives that will eventually lead to a routing and/or a scheduling change in the sensor nodes.

B. Implementation details

In the ONOS controller, the *Whisper Provider* abstracts the network information coming from the 6TiSCH network to the ONOS core, adding and updating links, devices, hosts and intents. The *Whisper Protocol* ONOS component feeds the *Whisper Provider* with information coming from the actual WSB through the REST API.

From the point of view of ONOS, sensor nodes are treated as special wireless switches. However, since sensor nodes are IPv6-enabled, the *Whisper Provider* adds virtual hosts to each sensor node to assign them IPv6 addresses. This way path Intents can be end-to-end created from sensor nodes to hosts. Intents are controlled with a new weight metric that is

[1] The complete set of messages and the detailed byte-level format is available at https://github.com/imec-idlab/whisper-repository

WSB Segment	Dir	Name	Arguments	Output
Orchestrator Whisper	DL	ParentSwitch	NodeA,NodeB	ResponseCode
	DL	AddCell	NodeA,NodeB,Cells*	ResponseCode,CellList
	DL	DeleteCell	NodeA,NodeB,Cells*,Clear*	ResponseCode
	UL	NetworkUpdate	NodeID,Topology*,LinkCost*,Schedules*	none
Whisper 6TiSCH	DL	SwitchRemote	TargetNode,FirstHop,Rank,ReliableSwitch*	ResponseCode
	DL	SwitchImpersonate	WhisperNode,TargetNode,ImpID,Rank,ReliableSwitch*	ResponseCode
	DL	PropagateRank	Rank,NodeID	ResponseCode
	DL	6PRequest	NodeA,NodeB,Cells*	ResponseCode,CellList
	DL	UpdateSolicitation	NodeID,Ranks*,Topology*,LinkPDR*,Schedules*,State*	ResponseCode
	UL	UpdateReport	NodeID,Ranks*,Topology*,LinkPDR*,Schedules*,State*	none

TABLE I: Whisper Southbound messages. DL denotes Downlink and UL denotes Uplink. Optional fields are denoted with *.

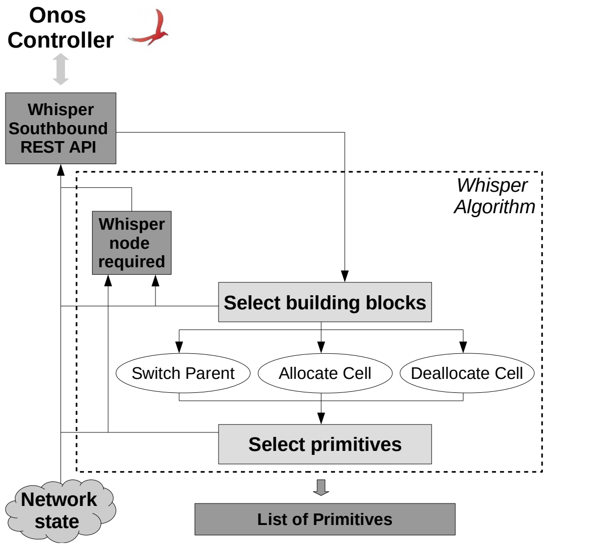


Fig. 6: Whisper controller architecture. Commands from ONOS arriving from the south-bound interface are translated to 6TiSCH compatible primitives to exert the control. Monitoring information is aggregated and reported to the orchestrator.

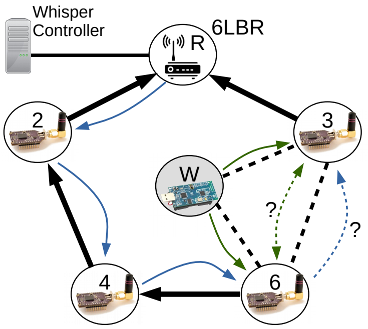


Fig. 7: Checking if a link exists: **a)** source route pings from the controller (blue), **b)** probing from the Whisper node (green).

assigned to the links as *Annotations* (e.g., in order to re-route an Intent through a certain path, a high value will be assigned to one of the links in that path). Also, at the application level, we have extended the ONOS GUI with 6TiSCH-related components and dashboards (augmented topology, wireless nodes, schedules, etc.), and a new Whisper command line interface has been included.

The Whisper controllers are deployed by the 6TiSCH network and are implemented (optionally) inside OpenVisualizer, a tool to interconnect a 6TiSCH network into the Internet. OpenVisualizer is included in the OpenWSN project [29], which is currently the most up-to-date implementation of 6TiSCH. In order to communicate with ONOS, the REST API is used to send reports and receive commands. In the Whisper controllers lays the intelligence for translating a network abstracted command to an actual set of Whisper commands (primitives) that achieves the desired change on the routing or the scheduling plane. This is done by following the building blocks depicted in Figure 6 and applying the algorithms described in [6]. However, in order to perform any effective change in the network, information about the 6TiSCH network has to be actively gathered. Since we assume nodes do not have any monitoring mechanism built-in (e.g., can be legacy devices), non-invasive network monitoring is not a trivial task. We further detail how to do this in the following Section IV-C.

C. Monitoring the 6TiSCH network through Whispering

While the ONOS controller only requires high level network information (e.g., topology, schedules, etc.), Whisper controllers require to have detailed information about the state of the network such as Ranks, 6P sequence numbers, DODAG versions, etc. Some network information is available at the controller, for example, the DODAG topology is directly obtained from the Destination Advertisement Object (DAOs) messages, which arrive to the Whisper controller through the 6LBR to build downlink routes. However the actual physical topology between neighbors is required to be known before any route change. Otherwise, the orchestrator would not even know if a change is actually possible. To check if a link exists, the Whisper controller can choose between using source routed pings or through 6P/RPL probing from a Whisper node (see node W in Figure 7). 6P/RPL probing consists in making the Whisper node impersonate one of the nodes to send non-effective 6P commands (COUNT, LIST, or SIGNAL) for the case of 6P probing, or unicast DODAG Information Solicitation (DIS) messages for the case of RPL probing, as shown in Figure 7 where node W impersonates node 6 to send such messages to node 3. If there is any response from the impersonated node, it means the link exists. Source routed pings and RPL probing are the simplest and safest options due to uncaptured errors in 6P probing could lead to 6P SeqNum mismatch between the nodes. Continuous probing is used to estimate the PDR of a given link.

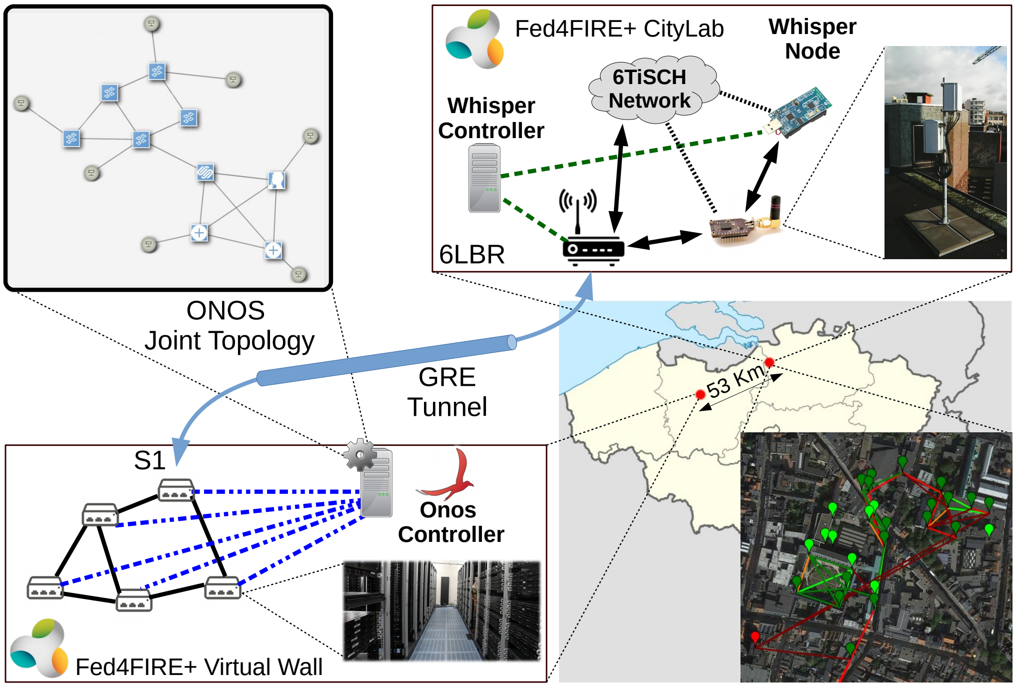


Fig. 8: Architecture of the multi-domain network deployed in two different Fed4FIRE+ testbeds to provide end to end control.

The 6P SeqNum can also be monitored to both obtain scheduling information (LIST) and perform scheduling transactions (ADD/DELETE/CLEAR). Although nor strictly necessary since SeqNum reset can be forced, it is advisable to track the SeqNum at any node at the Whisper nodes or the 6LBR (root) to reduce overhead. On the other hand, Rank values can be obtained either sniffing DIOs in the root or in the Whisper nodes or by emitting DIS messages to specific nodes out-of-range. Even though Ranks are initially estimated at the Whisper controller using the DODAG information and the default RPL objective function, having the actual ranks helps to more accurately know the network state and roughly estimate the link PDRs. Finally, a number of UpdateReports from different Whisper nodes and 6LBRs is periodically aggregated at the Whisper controllers. However only high-level information will be included in one single *NetworkUpdate* message destined to the ONOS core.

V. Deploying the System

Let us assume now Scenario 2 as illustrated in Figure 9, where, as before in Scenario 1 (Section II), sensor nodes measure the state of critical assets (e.g., vibration level of a machine). In this case, sensor nodes are centralizing data in node 2 before sending it to the core, to perform data aggregation or fusion techniques (e.g., to filter redundant data). In the core, data is first analyzed in real-time in a Deep Packet Inspection (DPI) box located in S4 and stored in H. Imagine now that node 4 detects an anomalous reading. What a network operator may want here is to closely monitor that anomalous node 4 with as lowest latency as possible (skipping the DPI since involves a longer path), to avoid future damage in the machine in case readings get over a threshold. Subsequently, node 7's next hop has to change to continue aggregating its readings, letting node 4 isolated. Therefore, a high-level orchestrator needs to reconfigure the network both in the wired and wireless segment to match this new requirement, for example as shown in Figure 9 b).

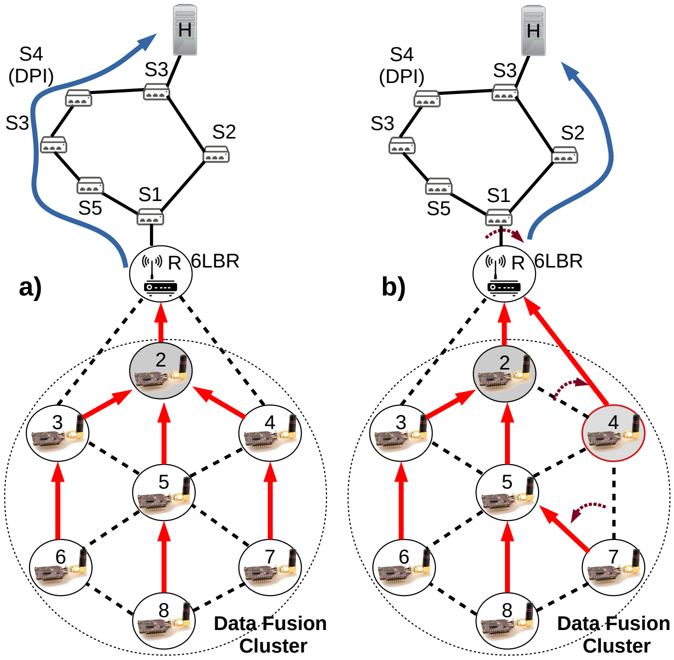


Fig. 9: In Scenario 2, a) shows the situation when the 6LoWPAN aggregates data at node 2. In b), low latency is required and network is end-to-end reconfigured to achieve it.

Within this section we show a realistic, complex deployment of Scenario 1 and 2 in two large-scale open Fed4FIRE+ [30]

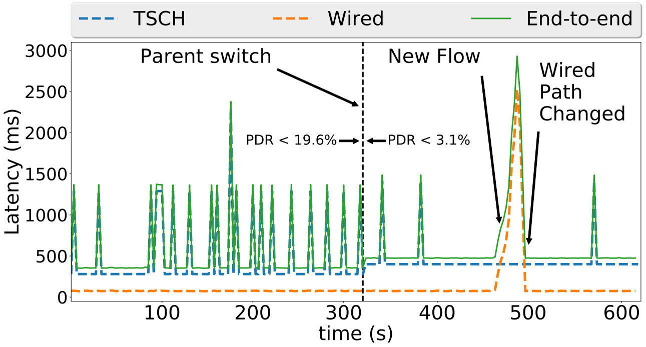


Fig. 10: Evolution of latency for Scenario 1. First, data from node 6 is sent directly to the root with some PHY drops, causing retransmissions. After high-reliability becomes a requirement, traffic is re-routed using higher quality links and avoiding saturated links that could cause drops.

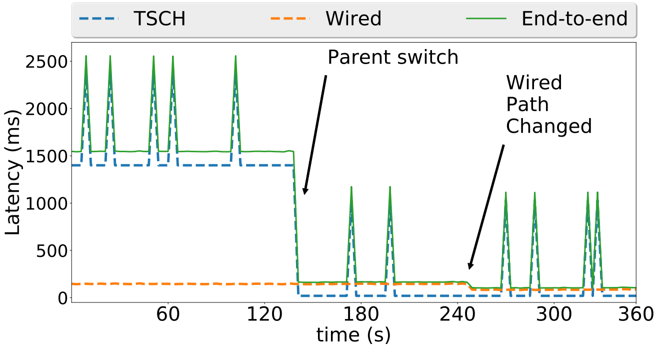


Fig. 11: Evolution of latency for Scenario 2. Data from node 4 is re-routed through the path with lowest latency. For the wireless segment, also cells are scheduled to minimize latency.

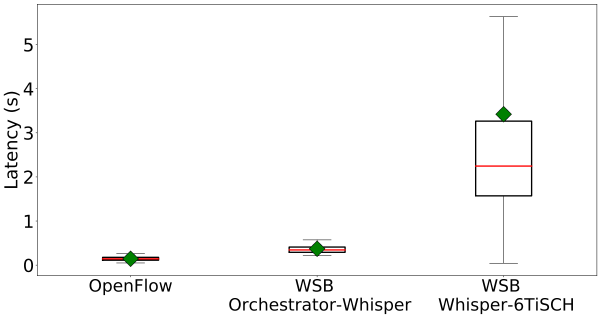


Fig. 12: Distribution of latencies for the control traffic. The average values (green diamond) are 0.14, 0.37 and 3.42s respectively. Median values (red line) are 0.14, 0.34 and 2.24s.

federated testbed facilities[2]. The whole set-up is depicted in Figure 8. On one hand, the Virtual Wall testbed[3], located in Ghent, Belgium, is used to deploy the OpenFlow-enabled wired network. The network consists of a set of barebone switches that run OpenVSwitch 2.9 and are operated through OpenFlow. An additional server is used as orchestrator, where a Whisper-enabled ONOS 2.1.0 is deployed to manage both wireless and wired networks. On the other hand, we use the CityLab smartcity testbed[4] [31], located in Antwerp, Belgium, to deploy the 6TiSCH wireless network remotely controlled from Ghent. This network is formed by OpenMotes-CC2538 nodes [32] running OpenWSN Release 1.24. Since the distance between the two testbeds is 53 Km, the connectivity between them is done by a GRE tunnel between the 6LBR in CityLab and the gateway node S1 in the Virtual Wall. In this node, the *gretap* interface of the tunnel is added to the Open vSwitch bridge to be directly managed by the ONOS controller.

As previously presented, Scenario 1 topology is deployed as depicted in Figure 1. This use case requires to select the most reliable links in both the wireless and the wired segment for the target flow. Figure 10 shows the evolution of latency for data sent from node 6 to host H at a rate of 1 pkt every 3s. Although PHY drops ($\sim$ 19.6%) exist during the power saving regime, these drops do not actually count as end-to-end packet loss since retransmissions over different channels ensure a successful delivery. However, they cause peaks in latency of about 1 $\sim$ 2 s as observed in Figure 10. If a packet is dropped, it will be re-transmitted in the next TSCH frame, after 101 *slots* x 10 *ms timeslot* = 1.01 s (or 2.02 s if two retransmissions occur).

After $t \simeq 325$, the parent of node 6 is changed to ensure the highest link reliability (PHY drops are now $\sim$ 3%), and thus, retransmissions are reduced. However, at $t \simeq 475$, the presence of a saturation flow (from host A to Host B) in the wired network, compromises the reliability requirement. This flow could induce real drops when the queues become full. For this reason, after the new flow enters, ONOS triggers a path change in the wired network that reroutes the target flow to prevent any packet drop, as depicted in the Figure 10.

Similarly, we also presents results on the Scenario 2. In this case, the topology is changed as depicted in Figure 9 in order to ensure the lowest low-latency in node 4. The evolution of the latency of node 4 is shown in Figure 11. Around $t \simeq 140$, node 4 changes his parent to reduce the TSCH latency. This is achieved not only just by reducing the number of hops to the root but also carefully choosing the cells that produce lower latency (e.g., placing the new cells just after the packet is enqueued). Similarly, at $t \simeq 240$, ONOS changes the route in the wired segment as well in order to skip the DPI and reduce the number of hops, and eventually, minimizing the end-to-end latency of node 4 incoming data.

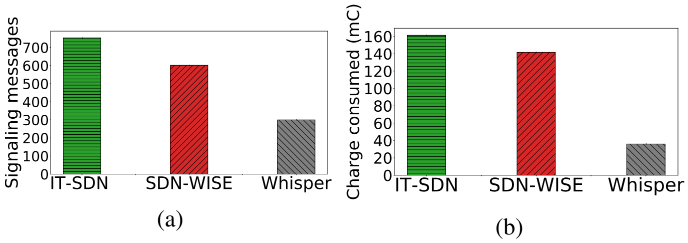
For both scenarios, no modifications have been done in the firmware of the already deployed nodes. Also, cells have been pre-allocated beforehand by Whisper to avoid packet drop during the parent change. All discovering and monitoring tasks are done in the background since the network bootstrap. This allows the local Whisper controller to obtain the real topology and gather routing (e.g., preferred parents and Ranks) and scheduling information (e.g., used cells).

We have also measured the latency of the complete set up for the control traffic. To do so, we perform the path changes described in the Scenario 2 scenario during 5 hours at a rate of 1 time per minute. Figure 12 shows the distribution of

[2] All used code and experiment traces obtained are publicly available at https://github.com/imec-idlab/whisper-repository

[3] https://www.fed4fire.eu/testbeds/virtual-wall/

[4] https://www.fed4fire.eu/testbeds/citylab/



(a)

(b)

Fig. 13: For each solution, a) required number of control messages, and b) energy consumed only in signaling overhead.

latencies for each segment. This is calculated as the time elapsed between the time when the command is sent and the arrival time of the first packet that follows the new rule. For the case of the WSB protocol, the total time is divided in two: first, the communication between ONOS and the Whisper controller (WSB Orchestrator-Whisper) and second, the actual message injection in the 6TiSCH network done by the Whisper controller (WSB Whisper-6TiSCH). The time used in the Whisper-6TiSCH segment is expectedly higher due to commands are delivered through the wireless medium (retransmissions may be needed) and can only be transmitted in specific cells (minimal cell or other dedicated cells).

Finally we have analyzed, for the wireless segment in the former test, the impact of using Whisper in terms of number of signaling messages and energy consumption due to signaling overhead. We have also evaluated the equivalent required messages and related energy consumption for the SDN-on-IoT solutions IT-SDN and SDN-WISE. Figure 13 shows that Whisper requires less control messages to perform the path changes (and required scheduling modifications), and consequently, a lower energy consumption due to signaling overhead.

These tests illustrate the performance of the tested networks when a high-level controller jointly manages them, without the need to modify the firmware of the already deployed nodes. While in the wired segment control is exerted with standard Intent-based OpenFlow rules, the use of Whisper in the wireless segment allows ONOS to also monitor and manage the routing and scheduling of the 6TiSCH network.

VI. Conclusion

In this work we have introduced relevant examples that demonstrate the need for a central control that jointly manages the full end-to-end IoT-Wired domain. From these scenarios, we have presented a new higher-level solution that leverages Whisper to exert control in both the wired segments and the Industrial IoT domain.

In between a fully centralized SDN-on-IoT management solution and a traditional fully distributed one, Whisper seems to stay as a trade-off solution that has the robustness, scalability and low-overhead of distributed solutions and the flexibility and programmability of centralized ones, without the need of modifying the firmware of the nodes in an already deployed Industrial IoT network, and thus, no requiring an additional SDN-specific protocol.

We have also presented the framework that shifts the Whisper scope from the edge to the orchestrator, describing the new Whisper Southbound protocol that enables its integration with a generic controller. Finally, the presented solution has been validated by deploying it in real world large scale testbeds and by testing on them several Industrial IoT use cases that require flexible end-to-end control over a heterogeneous network domain.

The results point towards an effective holistic network management that can fulfill dynamic network requirements (e.g., reliability and low-latency). To the initially asked question of "*Can IoT networks be efficiently sofwarized jointly with core the network?*", these results, together with the presented use cases, make us confident to affirmatively answer that not only it is possible, but essential for current Industry 4.0 demands. Also, while we see this architecture as a promising alternative for future Industrial IoT deployments and therefore as an challenging research path, we believe that it is definitively interesting for IoT legacy deployments, which can benefit from a fully end-to-end upgraded network control with no modification in the already-deployed IoT network.

VII. Acknowledgement

This work has partially received funding from the European Union's Horizon 2020 Fed4FIRE+ project.

References

[1] E. Sisinni, A. Saifullah, S. Han, U. Jennehag, and M. Gidlund, "Industrial Internet of Things: Challenges, opportunities, and directions," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 11, 2018. [Online]. Available: 10.1109/TII.2018.2852491

[2] D. Dujovne, T. Watteyne, X. Vilajosana, and P. Thubert, "6TiSCH: Deterministic IP-enabled Industrial Internet (of things)," *IEEE Communications Magazine*, vol. 52, no. 12, pp. 36–41, 2014. [Online]. Available: 10.1109/MCOM.2014.6979984

[3] T. Watteyne, L. Doherty, J. Simon, and K. Pister, "Technical overview of SmartMesh IP," in *2013 Seventh International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS)*. IEEE. [Online]. Available: 10.1109/IMIS.2013.97

[4] Emerson, "Industrial Wireless Technology." [Online]. Available: https://www.emerson.com/en-us/expertise/automation/industrial-internet-things/pervasive-sensing-solutions/wireless-technology

[5] S. Bera, S. Misra, and A. V. Vasilakos, "Software-defined networking for Internet of Things: A survey," *IEEE Internet of Things Journal*, vol. 4, no. 6, pp. 1994–2008, 2017. [Online]. Available: 10.1109/JIOT.2017.2746186

[6] E. Municio, J. Marquez-Barja, S. Latré, and S. Vissicchio, "Whisper: Programmable and Flexible Control on Industrial IoT Networks," *Sensors*, vol. 18, no. 11, 2018. [Online]. Available: http://www.mdpi.com/1424-8220/18/11/4048

[7] M. Wollschlaeger, T. Sauter, and J. Jasperneite, "The future of Industrial Communication: Automation networks in the era of the Internet of Things and Industry 4.0," *IEEE industrial electronics magazine*, vol. 11, no. 1, pp. 17–27, 2017. [Online]. Available: 10.1109/MIE.2017.2649104

[8] P. Ferrari, A. Flammini, S. Rinaldi, E. Sisinni, D. Maffei, and M. Malara, "Impact of Quality of Service on Cloud based Industrial IoT applications with OPC UA," *Electronics*, vol. 7, no. 7, p. 109, 2018.

[9] D. Bruckner, M.-P. Stănică, R. Blair, S. Schriegel, S. Kehrer, M. Seewald, and T. Sauter, "An introduction to OPC UA TSN for Industrial Communication Systems," *Proceedings of the IEEE*, vol. 107, no. 6, pp. 1121–1131, 2019. [Online]. Available: 10.1109/JPROC.2018.2888703

[10] M. Aazam, S. Zeadally, and K. A. Harras, "Deploying Fog Computing in Industrial Internet of Things and Industry 4.0," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4674–4682, 2018. [Online]. Available: 10.1109/TII.2018.2855198

[11] J. Graham-Cumming, “Details of the Cloudflare outage on July 2, 2019.” [Online]. Available: https://blog.cloudflare.com/details-of-the-cloudflare-outage-on-july-2-2019/

[12] P. Thubert, M. R. Palattella, and T. Engel, “6TiSCH centralized scheduling: When SDN meet IoT,” in *Standards for Communications and Networking (CSCN), 2015 IEEE Conference on*. IEEE, 2015.

[13] L. Galluccio, S. Milardo, G. Morabito, and S. Palazzo, “SDN-WISE: Design, prototyping and experimentation of a stateful SDN solution for WIreless SEnsor networks,” in *2015 IEEE Conference on Computer Communications (INFOCOM)*. IEEE, 2015, pp. 513–521. [Online]. Available: 10.1109/INFOCOM.2015.7218418

[14] B. T. De Oliveira, L. B. Gabriel, and C. B. Margi, “TinySDN: Enabling multiple controllers for software-defined wireless sensor networks,” *IEEE Latin America Transactions*, vol. 13, no. 11, pp. 3690–3696, 2015.

[15] M. Baddeley, R. Nejabati, G. Oikonomou, M. Sooriyabandara, and D. Simeonidou, “Evolving SDN for Low-Power IoT Networks,” in *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*. IEEE, 2018, pp. 71–79.

[16] A. Dunkels, O. Schmidt, N. Finne, J. Eriksson, F. Österlind, and N. T. M. Durvy, “The Contiki OS: The operating system for the Internet of Things,” 2011. [Online]. Available: http://www.contikios.org

[17] T. Chang, M. Vučinić, X. Vilajosana, S. Duquennoy, and D. Dujovne, “6TiSCH Minimal Scheduling Function (MSF),” Internet Engineering Task Force, Internet-Draft draft-ietf-6tisch-msf-05, Jul. 2019, work in Progress.

[18] A. Greenberg, G. Hjalmtysson, D. A. Maltz, A. Myers, J. Rexford, G. Xie, H. Yan, J. Zhan, and H. Zhang, “A clean slate 4D approach to network control and management,” *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 5, pp. 41–54, 2005. [Online]. Available: https://doi.org/10.1145/1096536.1096541

[19] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker, “NOX: towards an operating system for networks,” *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 3, 2008. [Online]. Available: https://doi.org/10.1145/1384609.1384625

[20] J. Medved, R. Varga, A. Tkacik, and K. Gray, “Opendaylight: Towards a model-driven sdn controller architecture,” in *Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014*. IEEE, 2014, pp. 1–6.

[21] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, “Openflow: enabling innovation in campus networks,” *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[22] R. Enns, M. Bjorklund, and J. Schoenwaelder, “Network configuration protocol (netconf),” *Network*, 2011. [Online]. Available: http://www.hjp.at/doc/rfc/rfc6241.html

[23] P. A. working group, “P4 runtime.” [Online]. Available: https://p4.org/p4-runtime

[24] P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O’Connor, P. Radoslavov, W. Snow *et al.*, “ONOS: towards an open, distributed SDN OS,” in *Proceedings of the third workshop on Hot topics in software defined networking*. ACM, 2014, pp. 1–6. [Online]. Available: https://doi.org/10.1145/2620728.2620744

[25] L. Mamushiane, A. Lysko, and S. Dlamini, “A comparative evaluation of the performance of popular SDN controllers,” in *2018 Wireless Days (WD)*. IEEE, 2018, pp. 54–59.

[26] A.-C. G. Anadiotis, L. Galluccio, S. Milardo, G. Morabito, and S. Palazzo, “Towards a software-defined Network Operating System for the IoT,” in *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*. IEEE, 2015, pp. 579–584.

[27] S. Fichera, M. Gharbaoui, P. Castoldi, B. Martini, and A. Manzalini, “On experimenting 5G: Testbed set-up for SDN orchestration across network cloud and IoT domains,” in *2017 IEEE Conference on Network Softwarization (NetSoft)*. IEEE, 2017, pp. 1–6.

[28] Z. Shelby, K. Hartke, and C. Bormann, “The constrained application protocol (CoAP),” 2014.

[29] T. Watteyne, X. Vilajosana, B. Kerkez, F. Chraim, K. Weekly, Q. Wang, S. Glaser, and K. Pister, “OpenWSN: a standards-based low-power wireless development environment,” *Transactions on Emerging Telecommunications Technologies*, vol. 23, no. 5, pp. 480–493, 2012.

[30] B. Vermeulen, W. Van de Meerssche, and T. Walcarius, “jFed toolkit, Fed4FIRE, federation,” in *GENI Engineering Conference*, vol. 19, 2014.

[31] J. Struye, B. Braem, S. Latré, and J. Marquez-Barja, “Citylab: A flexible large-scale multi-technology wireless smartcity testbed,” in *Proceedings of the 27th European Conference on Networks and Communications (EUCNC), 18-21 June 2018, Ljubljana, Slovenia*, 2018, pp. 374–375.

[32] X. Vilajosana, P. Tuset, T. Watteyne, and K. Pister, “OpenMote: open-source prototyping platform for the industrial IoT,” in *International Conference on Ad Hoc Networks*. Springer, 2015, pp. 211–222.

Esteban Municio received a Bsc+MSc degree of Telecommunication Engineering from the Madrid Polytechnic University (UPM) in 2013 and a MSc degree in Networks and Computer Systems from King Juan Carlos University (URJC) in 2014. He is obtaining his PhD in Computer Science from the University of Antwerp in February 2020. He is currently a PhD researcher at the IMEC - IDLab research group, in the Department of Mathematics and Computer Science of the University of Antwerp. He was and is involved in several European research projects such as TUCAN3G, FLEXNET and INTERCONNECT. His main research interests are: traffic engineering and SDN network programmability, flexible and programmable wireless networks and ultra-reliable Industrial IoT. He is also interested in heterogeneous backhaul networks, smart cities deployments, community networks and connectivity provision in rural environments.

Steven Latré received the Master of Science degree in computer science and the Ph.D. degree in computer science engineering from Ghent University, Belgium, with the thesis title ”Autonomic Quality of Experience Management of Multimedia Services”. He is currently a Professor at the University of Antwerp, and Director at the imec research centre. He is leading the IDLab Antwerp Research Group (100+ members), which is performing applied and fundamental research in the area of communication networks and distributed intelligence. He has authored or coauthored over 100 papers published in international journals or in the proceedings of international conferences. His personal research interests are in the domain of machine learning and its application to wireless network optimization. He was a recipient of the IEEE COMSOC Award for Best Ph.D. in network and service management in 2012 and the IEEE NOMS Young Professional Award in 2014. He is a member of the Young Academy Belgium.

Johann M. Marquez-Barja currently is an Associate Professor at University of Antwerpen - imec. He is the Head of the Wireless Cluster at imec IDLab UAntwerp. He was and is involved in several European research projects, being Principal and Co-Principal Investigator for many projects. He is an ACM member, and a Senior member of the IEEE Communications Society as well as the IEEE Education Society, where he participates in the Standards Committee. His main research interests are: 5G advanced architectures including edge computing; flexible and programmable future end-to-end networks; IoT communications and applications. He is also interested in vehicular communications, mobility, and smart cities deployments. Prof. Marquez-Barja is co-leading the Citylab Smart City testbed, part of the City of Things programme, located in Antwerpen, Belgium. Prof. Marquez-Barja has been given several keynotes and invited talks in different major events, as well as received 25 awards in his career so far, and co-authored more than 100 articles. He is also serving as Editor and Guest editor for different International Journals, as well as participating in several Technical Programme and Organizing Committees for several worldwide conferences/congresses.