Hindawi
Wireless Communications and Mobile Computing
Volume 2018, Article ID 2946575, 12 pages
https://doi.org/10.1155/2018/2946575

Research Article

Software Architecture Solution Based on SDN for an Industrial IoT Scenario

José L. Romero-Gázquez [1] **and M. Victoria Bueno-Delgado** [1,2]

[1] *Universidad Politécnica de Cartagena, Grupo en Ingeniería en Redes de Telecomunicación, Plaza del Hospital 1, Edif. Antigones, 30202, Cartagena, Spain*
[2] *E-lighthouse Network Solutions S.L., 30203, Cartagena, Spain*

Correspondence should be addressed to José L. Romero-Gázquez; josel.romero@upct.es

Received 12 April 2018; Revised 20 July 2018; Accepted 5 September 2018; Published 27 September 2018

Guest Editor: Muhammad Imran

The Industry 4.0 (I4.0) adoption comprises the change of traditional factories into *smart* using the ICTs. The goal is to monitor processes, objects, machinery, and workers in order to have real-time knowledge about what is going on in the factory and for achieving an efficient data collection, management, and decision-making that help improve the businesses in terms of product quality, productivity, and efficiency. Internet of Things (IoT) will have an important role in the I4.0 adoption because future smart factories are expected to rely on IoT infrastructures composed of constellations of hundreds or thousands of sensor devices spread all over the industrial facilities. However, some problems could arise in the massive IoT deployment in a medium-high factory: thousands of IoT devices to cope from different technologies and vendors could mean dozens of vendor tools and user interfaces to manage them. Moreover, the heterogeneity of IoT devices could entail different communication protocols, languages, and data formats, which can result in lack of interoperability. On the other hand, conventional IT networks and industrial machinery are expected to be managed together with the IoT infrastructure, maybe using a tool or a set of tools, for *orchestrating* the whole smart factory. This work meets these challenges presenting an open-source software architecture solution based on OpenDaylight (ODL), a Software Defined Network (SDN) controller, for orchestrating an industrial IoT scenario. This work is addressed by shedding light on critical aspects from the SDN controller architectural choices, to specific IoT interfaces and the difficulties for covering the wide range of communication protocols, popular in industrial contexts. Such a global view of the process gives light to practical difficulties appearing in introducing SDN in industrial contexts, providing an open-source architecture solution that guarantees devices and networks interoperability and scalability, breaking the vendor lock-in barriers and providing a vendor-agnostic solution for orchestrating all actor of an I4.0 smart factory.

1. Introduction

The European Union (EU) is moving towards what is already considered the fourth industrial revolution, the Industry 4.0 (I4.0). The aim of I4.0 is to transform the conventional factories into smart factories using the Information and Communication Technologies (ICTs). These add flexibility in manufacturing, mass customization, improve the quality of products and processes, and increase the productivity. A subset of ICTs, the Key Enabling Technologies (KET), have been proposed as drivers to carry on this change: the use of sensors, actuators, wireless communications, next generation networks, robots, additive manufacturing, cloud computing, big data and cybersecurity, among others [1]. In this new technological scenario many strategic points (products, processes, workplaces, workers, etc.) will be monitored by using different technologies and devices. These will be wired or wireless connected to the factory local network or to the Internet. One or more software tools will be in charge of collecting, storing, filtering, and managing the enterprise data. This scenario will be carried out by setting an Internet of Things (IoT) infrastructure [2, 3]. Hundreds or thousands IoT devices will be spread in the factory collecting data at real-time about industrial processes, machinery, and workers performance. The goal is to achieve an efficient factory management and decision-making to improve the

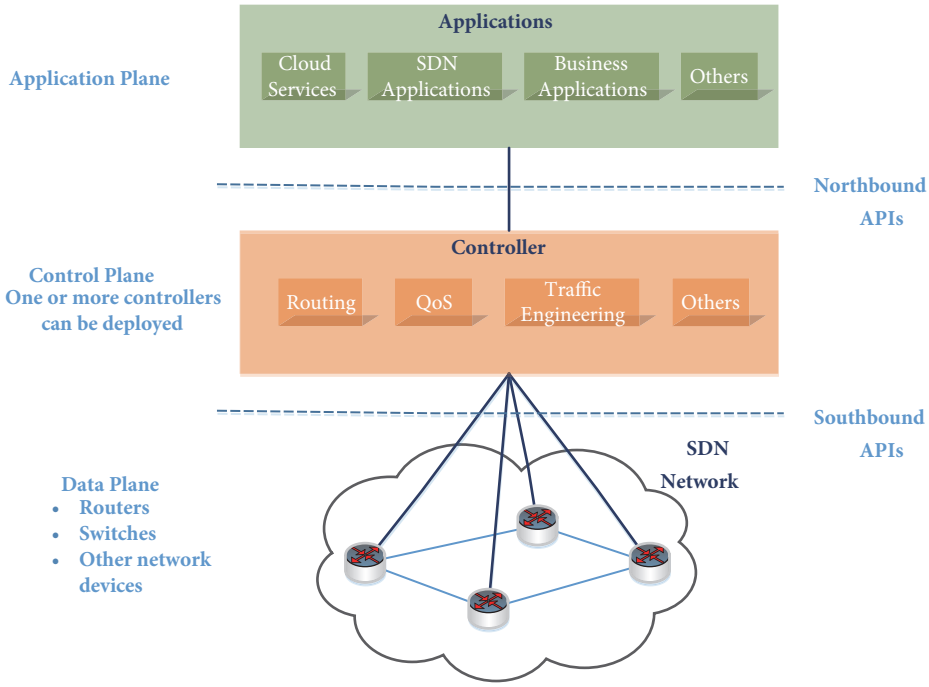

Figure 1: Three layered-based Software Defined Network architecture.

business in terms of product quality, productivity, and efficiency.

The I4.0 adoption in the above scenario entails different problems for a medium-high factory:

(i) Thousands of IoT devices to cope, from different vendors and with different technologies, could mean dozens of vendor tools and user interfaces to manage them.

(ii) The heterogeneity of IoT devices could entail different communication protocols, languages and data formats, among others, which can result in lack of interoperability.

(iii) Conventional IT networks and industrial machinery will coexist with the new IoT infrastructures, and all will be "connected" to achieve the full I4.0 adoption.

(iv) The data traffic in the industrial network infrastructure deployed could significantly increase, deriving in undesirable packet delays and network congestion. The data traffic should be routed efficiently to avoid it.

The two former could be managed by using a local or cloud-based IoT platform [4, 5] but the two latter would remain partially or totally unsolved. Some works in the recent scientific literature have proposed some software solutions to deal with these problems (see Section 2). However, they do not deal with heterogeneous scenarios where noncompliant IoT devices or industrial machinery are operating. This work tackles these challenges, by proposing a single open-source software architecture solution based on a layered-based Software Defined Network (SDN) architecture (see Figure 1).

SDN was firstly designed to orchestrate IT networks, but nowadays some SDN controllers include plugins to connect in the southbound with IoT devices and networks. The proposed software uses OpenDaylight (ODL (https://www.opendaylight.org/)), a SDN controller with a dedicated IoT plugin called IoTDM that manages and stores data generated by IoT based devices according to oneM2M (http://www.onem2m.org) standard. This is required to provide a standardized interface to manage and interact with user-applications. Since IoTDM by default only understands native JSON/XML data formats, this work breaks the lack of interoperability with noncompliant industrial devices by providing a software architecture solution that guarantees interoperability among these devices/technologies and the IoTDM plugin in the southbound. Finally, ODL has to transfer the data from the southbound to the northbound and vice versa. Here another software entity is needed because in the southbound IoTDM manages data with a native JSON/XML format, and applications in the northbound could work with a generic JSON/XML format. This work proposes the use of a system of plugins based on oneM2M standard. This ensures agile management, collection, and presentation to the final user applications.

The software architecture proposed in this work gives light to the practical difficulties of adopting I4.0 to medium-high factories, proposing a modular architecture solution based on pre-existing open-source software. Its modularity allows anyone to develop new plugins for those technologies involved in any industrial scenario not considered in this work. The solution is also vendor-agnostic, breaking the vendor lock-in barriers, and it is scalable in terms of amount of devices to manage and technologies to support, because ODL permits to create a cluster of controllers running as

a single entity to allow fast scaling, high availability, data persistence, and fault tolerance.

Finally, the solution also exploits some of the SDN benefits:

(i) The management of tasks is automated and isolated from the complexity of the physical infrastructure through easy-to-use interfaces.

(ii) New services and applications could be provided in short time; in addition the IT administrators have the possibility of programming network functionalities and services as required, eliminating dependence on hardware manufacturers.

(iii) The network routing policies can be configured and managed for the whole network, using the same software solution, monitoring the network reliability, safety, and efficiency.

(iv) The user experience improves because the applications can exploit the centralized information about the state of the network and data collected, making it possible to react in real time and to carry out modifications that can improve the network performance.

(v) The operating costs of the industrial network management could be significantly reduced.

An Industrial IoT (IIoT) pilot scenario has been deployed to test the software architecture proposed. An application in the northbound has been implemented for testing some management tasks: add, remove, configure, and manage different IIoT devices, commonly used in industrial context. The scenario has been deployed with a single instance of ODL controller, giving the possibility of setting up a second ODL controller as backup to address risk situations such as high network latency, bottleneck, cybersecurity attacks, or faults.

Note that this work gives light to the practical difficulties appearing in introducing SDN in industrial contexts. It provides light to critical aspects from the controller architectural choices, to specific IoT interfaces and the difficulties for covering different communication protocols popular in industrial contexts. To do this, the paper is organized as follows: Section 2 reviews the related work. Section 3 summarizes the current IoT platforms and explains the reasons to use ODL and IoTDM in industrial scenarios. Section 4 reviews the most extended technologies and communication protocols in industrial facilities, remarking those that need a parser to be compatible with ODL and IoTDM. Section 5 presents the software architecture solution and Section 6 describes an IIoT pilot scenario testing the proposal. Finally Section 7 concludes.

2. Related Work

As we stated, it seems clear that the IIoT adoption is an unstoppable fact within the I4.0 framework. In a medium-high factory, it entails different problems that could be solved by using software solutions based on SDN, permitting the orchestration of IT, IoT, and IIoT networks. It seems right to focus the effort on the development of software services that allow (i) unifying the management of those heterogeneous networks given in an industrial scenario with a single and common modular software entity and (ii) providing interoperability in terms of technologies and devices popular in industrial contexts. Finally, (iii) the use of open standards should be mandatory, permitting to break the vendor lock-in problems and to create vendor-agnostic solutions.

Although the management of IoT and IIoT has received a lot of attention from the research community, to the best of the authors' knowledge there are only a few proposals where SDN is used to orchestrate these networks in an I4.0 scenario, but they do not provide all the features enumerated above. Some works solve the orchestration of IIoT networks without SDN, like in [6], where an IoT network is orchestrated by a network of *fog nodes*, which manage all layers of an IoT ecosystem to integrate different service modules into a complex topology. In [7] two architectures for an IoT networks orchestration are presented, using OSGi (https://www.osgi.org) and REST (https://restfulapi.net/), paying attention only to higher layers, not to physical connection. Moreover, universal standards for IoT like oneM2M are not taken into account, missing interoperability. Both proposals lack any single tool (industrial application and/or SDN orchestrator) to have a complete network knowledge and management. The same scenario is set in [8] where the main concepts involved in an IoT orchestrated network are explained, but without the use of SDN.

Other works present solutions based on SDN, but they do not look into to understand if they fulfil the requirements and solve the problem that I4.0 adoption entails in a medium-high factory. Some examples are in [9–17]. In [9] the authors analyse the IoT challenges that the network and IT infrastructures face, showing the Network Functions Virtualization (NFV) and SDN benefits from a network operator point of view. Work in [10] is focused on the fifth generation (5G) of mobile networks technology (5G) [11], applying SDN, NFV and IoT technologies. Works addressed in [12, 13] show an IoT architecture based on SDN, and [14] is a use case of [13] focused on 5G. As in the previous works, they do not provide a single solution where the IoT operations are managed on the SDN platform. In [15] the theory behind the use of SDN to manage an industrial network is explained, as well as in [16, 17], where the use of SDN for wireless networks (also called SDWN) and IoT is study in depth, highlighting some of the future research directions and open research issues based on the limitations of the existing SDN-based technologies.

At the time of writing this work, only a few works [18–20] provide a vendor agnostic IoT and SDN software architecture solution. However, they do not deal with heterogeneous scenarios where noncompliant IoT devices or industrial machinery are operating, and their solutions do not take into account oneM2M standard. In [18] a novel framework of IIoT with SDN and Edge Computation (EC) is proposed to address an optimal adaptive transmission routing mechanism. Open Mul (http://www.openmul.org/) is used as SDN controller. The authors do not look into how the architecture is designed, because the goal of the work is focused on the management of the network at upper layers. In [19] a SDN solution for

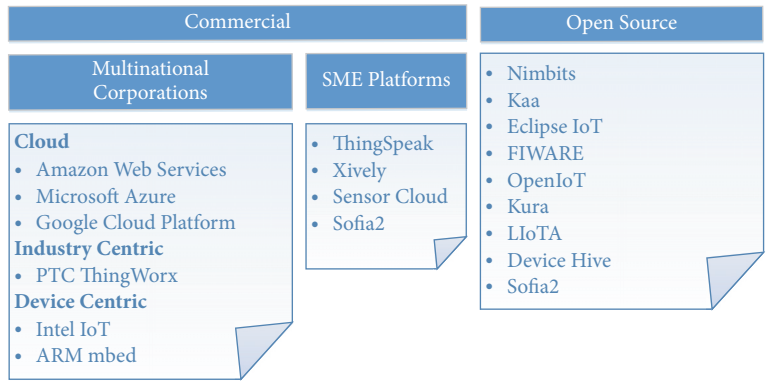

Figure 2: Leading IoT platforms.

IoT built with Open Network Operating System (ONOS (https://onosproject.org/)) as SDN orchestrator is proposed. The architecture is suitable for orchestrating end-to-end service chains deployed across heterogeneous SDN/NFV domains and define a related high-level and vendor-agnostic intent-based northbound interface (NBI). They propose a network management architecture based on the use of a dedicated controller for each type of network. In [20], the authors analyse the deployment of SDN in the Industry 4.0 paradigm, pointing out the three main problems and future issues to address for a fast and success implementation in industrial scenarios: data safety and system reliability, technology standardization and practical implementation. The authors suggest a solution to the above problems the development of a hybrid node device capable of handling with IT and IoT networks.

As a conclusion of this review, there are some works focused on explaining and developing SDN scenarios for conventional IT or IoT networks, but most of them are missing the interoperability of popular protocols and technologies of industrial contexts, the modularity, and the open-source benefits. This work is a contribution in this matter, providing light to critical aspects of SDN controller choice, IoT interfaces, and widely extended technologies and protocols of industrial contexts.

3. ODL as Industrial IoT Platform

There are many IoT platforms [4, 5] aiming at managing IoT networks, but it is difficult to find one that complies with the requirements of open source and heterogeneity for an Industrial IoT (IIoT) scenario. Figure 2 summarizes the main features of the most popular IoT platforms.

As the authors' knowledge most of open source IoT platforms enables interoperability, some of them through a multilanguage and multiprotocol system. They usually provide an SDK to develop applications. However, a real implementation in an industrial scenario with a wide range of industrial technologies requires big human resources to develop the functionalities required. In this point is where some companies offer their expensive commercial versions. This picture invited us to explore other software alternatives, like SDN, because nowadays some SDN platforms allow the integration of IoT services together with traditional human-based services, through a global orchestration of distributed cloud, heterogeneous IT, and IoT networks. In SDN (see Figure 1), the software entity *controller* (i) carries the huge amount of data generated by the IT and IoT networks to any distributed computing *node*, (ii) allocates computing and storage resources into distributed data centers, and (iii) processes the collected data to make decisions. In addition, SDN offers the possibility of network orchestration through the use of a cluster of controllers, which adds scalability to the management system, high availability, data persistence, and fault tolerance using a backup controller.

Among all available SDN controllers, this work has addressed a comparison from those that are open source and widely accepted by the scientific community [21, 22] to point out the one that meets the requirements of an IIoT scenario.

ONOS is an open source SDN controller written in Java with a distributed network operating system and good performance. It is composed by an extensible and modular platform and a set of applications. As southbound interfaces (SBI), ONOS supports OpenFlow, NETCONF, and PCEP. As northbound interfaces, it uses REST. Ryu (http://osrg.github.io/ryu/) is a component-based SDN controller with a set of predefined components written in Python, which can be modified, extended and composed to create a customized controller application. It supports different OpenFlow versions. Since it is written in Python, the network performance seems to be lower than other Java based SDN controllers, as stated in [21]. Floodlight (http://www.projectfloodlight.org/floodlight/) consists of a set of modules which provide service to other modules and/or to the control logic application through a simple Java or REST API. It can run on the top of Linux, Mac, and Windows OS. According to [23], Floodlight shows high performance in terms of packet loss. OpenDaylight is an open-source controller written in Java, with a high performance and widely supported by the networking industry. ODL includes IoT Data Broker (IoTDM) plugin, which opens the door to use ODL as a SDN platform for IoT networks management. IoTDM is a data-centric middleware that acts as an oneM2M compliant and enables authorized applications to retrieve IoT data uploaded by any device in the network. With IoTDM, ODL seems to be the most suitable candidate to manage an

IIoT. Table 1 summarizes a comparison of the main features provided by the SDN controllers reviewed.

4. Technologies and Communication Protocols in an Industrial IoT Scenario

In the previous section it was concluded that ODL together with IoTDM is our most suitable solution as IIoT platform. However, not all technologies and communication protocols coexisting in an IIoT scenario are compatible with ODL and IoTDM. Note that ODL and IoTDM works with XML/JSON data formats and HTTP, MQTT, or CoAP as communication protocols

In this section we review what technologies and communication protocols are widely used in an industrial facility, showing their data formats and compatibility with ODL and IoTDM. Table 2 summarized them. The goal is to identify the need of a *parser* in the southbound to enable communication between industrial technologies and ODL together with IoTDM. The technologies reviewed are the follows:

(i) Radio Frequency Identification (RFID) is one of the wireless technologies widely used in the industry as an alternative to the bar code. RFID uses the EPCglobal Class-1 Gen-2 standard as communication (and anticollision) protocol [24]. The data received and stored in the RFID reader from RFID tags can adopt XML format and can be sent via HTTP.

(ii) Bluetooth Low Energy (BLE (https://www.bluetooth.com)), the power-conserving variant of Bluetooth for personal area networks (PAN), is commonly used by Internet-connected machines and appliances. Its name also refers to its protocol, which is a full protocol stack. The BLE data format is not JSON or XML structure.

(iii) Modbus (http://www.modbus.org) is an industrial standard communication protocol, used for programmable logic controllers (PLCs), to transmit signals from instrumentation and control devices back to a main controller or data gathering system, and it is typically used for connecting a supervisory computer with a remote terminal unit (RTU) in supervisory control and data acquisition (SCADA) systems. There are different versions of Modbus protocol: Modbus RTU and Modbus ASCII for serial lines and Modbus TCP for Ethernet. The data format of Modbus is an Application Data Unit (ADU).

(iv) Controller Area Network (CAN or CAN Bus) [25] is a vehicle bus standard designed to allow electronic control units and devices to communicate with each other in applications without a host computer. Examples of devices working with CAN Bus are electronic control units, microcontrollers, devices, sensors, actuators, and other electronic components. CAN Bus is also a message based protocol, originally designed for multiplex electrical wiring within motor vehicles, but is used in the context of industrial applications. This standard generates DBC files, which are a proprietary format that describes the data over a CAN bus.

(v) Ethernet for Control Automation Technology (EtherCAT (https://www.ethercat.org)) is based on the CAN open protocol and on Ethernet but differs from network communications in being specifically optimized for industrial automation control, and its data format is EherCAT Slave Information (ESI).

(vi) Profibus and Profinet (https://us.profinet.com) are standards for fieldbus communications in the industrial automation. Profibus links controllers or control systems to decentralized field devices (sensors and actuators) on the field level and also enables consistent data exchange with higher ranking communication systems. Profinet connects devices, systems, and cells, facilitating faster, safer, less costly, and higher quality manufacturing. It easily integrates existing systems and equipment while bringing the richness of Ethernet down to the factory floor. Profinet communications are commonly TCP/IP based. Both Profinet and Profibus devices can connect to SCADA systems and IoT platforms through a gateway based on OPC (https://opcfoundation.org/), a communication standard for industrial telecommunication processes. OPC Unified Architecture (OPC UA) is the client-server architecture (see Figure 3) that serves as gateway to convert Profinet and Profibus data to proprietary XML/JSON data format. OPC UA supports real-time data communications, alarms, security features, etc. The OPC server is the data source and any application based on OPC standard can access to this server as OPC client to read and write on it. Software vendors only need to include OPC client capabilities in their products. Unfortunately, the industrial OPC software is not traditionally open source, and the companies are obeyed to spend a lot of money for locked-vendor OPC client-server solutions. Recently, some open-source OPC UA software solutions are available in [26].

5. Software Architecture Based on SDN and IoT for Industrial Iot Scenarios

Figure 1 is extended to Figure 4 to illustrate the software architecture proposed for an industrial I4.0 scenario if only open-source software is used. All devices (networks) are placed in the bottom, *orchestrated* by the SDN controller. In the bottom-left the IIoT network is placed, with IoT devices from different technologies (sensor motes, RFID, BLE) and industrial machinery working under Profinet, Ethercat, CAN Bus and Modbus. The IIoT network coexist with the conventional IT enterprise network, composed of routers, switches, PCs, printers, etc.

In the middle of Figure 4 a single controller is plotted, managing the IT and IIoT networks. A second controller is set as backup, to guarantee scalability and react to risk situations such as high computational load in the main controller, cybersecurity attacks, bottleneck in the IT or IIoT network,

Table 1: Comparison of SDN controllers.

SDN Controller	Language	Southbound	Northbound	Own IoT component	Modularity	Performance∗	Cluster capability
ONOS (https://wiki.onosproject.org/display/ONOS/System+Components)	Java	OpenFlow, OVSDB, NETCONF, ...	REST, RESTCONF, ...	Missing	High	High	Yes
Ryu (http://osrg.github.io/ryu/)	Python	OpenFlow, OVSDB library, NETCONF, ...	REST	Missing	Medium	Low	No
Floodlight (https://floodlight.atlassian.net/wiki/spaces/floodlight-controller/pages/1343549/Architecture)	Java	OpenFlow	REST	Missing	High	High	No
ODL (https://www.opendaylight.org/what-we-do/current-release/carbon)	Java	OpenFlow, PCEP NETCONF, OVSDB, BGP, ...	REST, RESTCONF, NETCONF, AMQP.	IoTDM	High	High	Yes

∗ Big amount of devices to orchestrate.

Table 2: Comparison of the most common technologies and communication protocols in an IIoT scenario.

Technology	Communication Protocol to	XML or JSON format	Parser in the southbound for SDN and IoTDM communication
IoT devices	HTTP, CoAP, or MQTT	Yes, XML, JSON	No
RFID	EPCglobal Class-1 Gen-2 (among RFID tags and reader) HTTP (between RFID reader and server)	Yes, XML	No
BLE	BLE	No, if OPC-UA is not used	Yes, if OPC-UA is not used
Modbus	Modbus RTU, Modbus ASCII, Modbus TCP	No∗	Yes∗
CAN Bus	CAN Bus	No∗	Yes∗
EtherCAT	CAN open-Ethernet	No∗	Yes∗
Profibus	CAN open-Ethernet	No∗	Yes∗
Profinet	TCP/IP	No∗	Yes∗

∗If OPC-UA is not used.

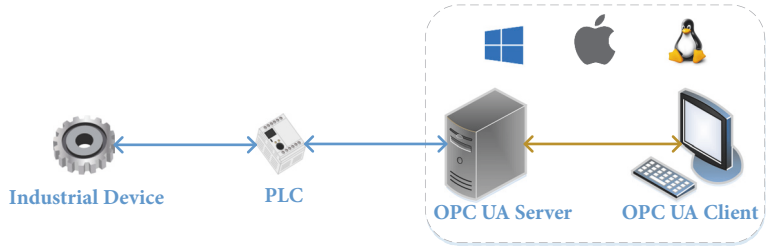

Figure 3: General OPC UA client/server communication.

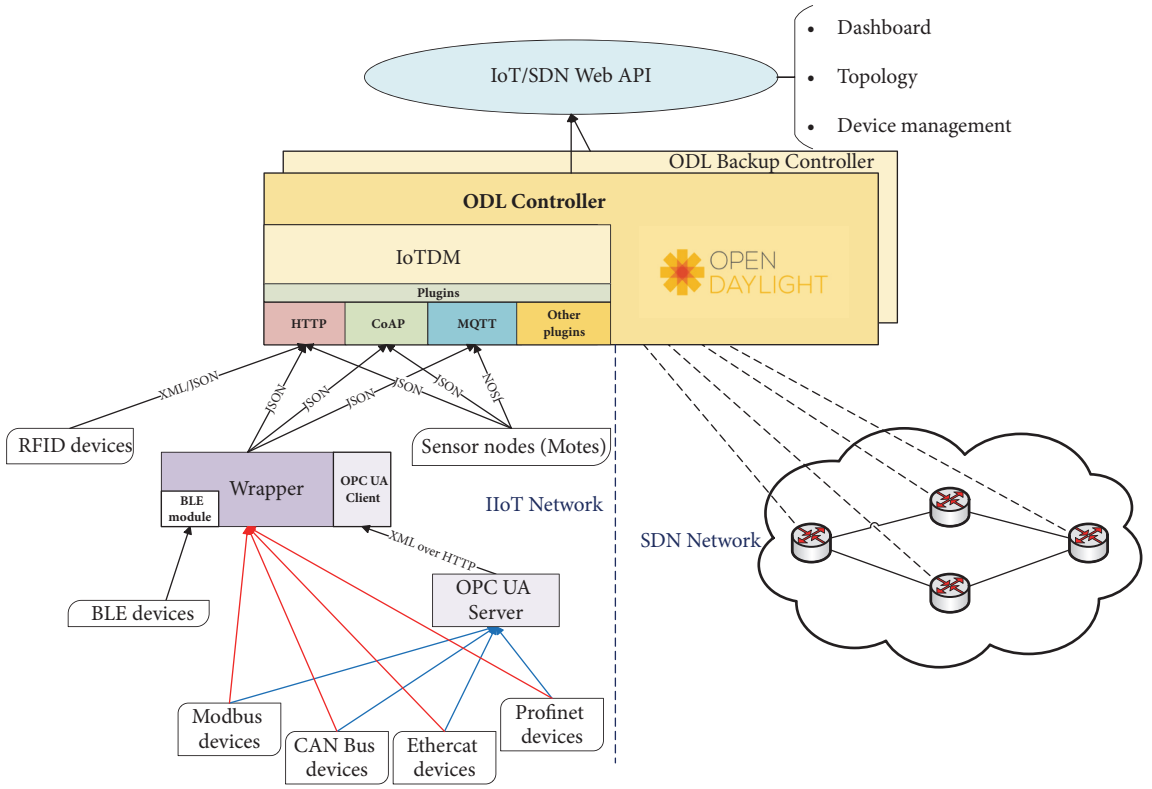

Figure 4: Industrial IoT software architecture solution.

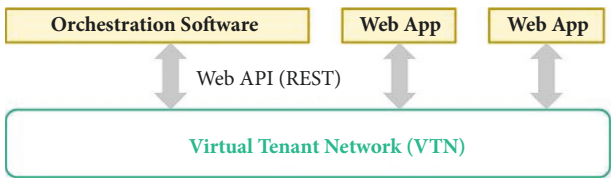

Figure 5: VTN orchestration Software.

etc. The decision comes from the fact that according to the work in [24], a single controller has more than enough powerful to manage the traffic of a set of medium-size IT and IoT networks in an industrial scenario. Authors in [27] show how a single NOX (https://github.com/noxrepo/nox-classic/wiki) controller can handle at least 30Kb flows per second with 10 ms of install delay. Hence, in this scenario, only two controllers are considered, but the number of controllers could vary because ODL allows a flexible deployment of controllers, depending on the needs. ODL uses a software functionality called Virtual Tenant Network (VTN) for the orchestration of a cluster of controllers (see Figure 5).

Another possibility of SDN controller deployment, not considered in this work, is the use of a dedicated SDN controller for each type of network: one controller for the IT network and one controller for the IoT network. This solution has some advantages such as modularity, more efficient management of applications in the northbound and southbound, or a decrease of bottleneck risk. However, this deployment shows some disadvantages: (i) more hardware, that is, at least two high-powerful computers to allocate each controller and two more for allocating backup controllers; (ii) to use a nonunified interface: the number of user interfaces will be equal to the number of controllers running in the deployment; (iii) to replicate NBI and SBI applications if they operate in both networks: the use of a single application for managing more than one network is very usual. For instance, NBI applications for visualization and device management with dashboard are needed in every network orchestration.

The deployment and the communication among all actors in the architecture proposed are explained in detail in the next subsections, from the physical to the application layer, that is, from the southbound to the northbound SDN architecture.

5.1. Southbound Layer. The southbound is the physical/logical connection between devices (networks) placed in the industrial scenario and the SDN controller (ODL and IoTDM), installed in a virtual or local machine. Three networks are identified: IT network, IoT network and noncompliant IoT networks, or industrial machinery networks. Conventional IT networks are directly connected to the ODL controller via wired or wireless connection, transmitting and receiving monitoring and management data with the widely used Openflow and NETCONF protocols and even with SNMP protocol if it is required, because ODL has a plugin to support it. The IoT network, composed of Wireless Sensor Networks (WSN) or a set of sensor nodes, is connected to ODL through IoTDM via wired or wireless connection (LoRa (https://www.lora-alliance.org/) and Sigfox (https://www.sigfox.com/) coud be used as wireless network). Other networks noncompliant with the communication protocols HTTP, CoAP, or MQTT and the data formats JSON or XML will need a gateway or *wrapper* to communicate with ODL through IoTDM.

RFID is compatible with IoTDM. RFID readers are the gateways of the RFID network deployments, sending and receiving the data to ODL and IoTDM in JSON or XML format, through HTTP. Most of IoT sensors (motes) in the market are also compatible with IoTDM. Motes acting as gateways in the sensor network deployments store and send data collected directly to ODL through IoTDM in JSON format using CoAP or MQTT protocol. TelosB, TinyOS, Beagle Bone, or Tmote Sky use CoAP while Raspberry Pi and Arduino can use both.

On the other hand, those devices operating with BLE need a *wrapper* to communicate with IoTDM. The wrapper is a SW+HW entity that can be implemented in the same machine where ODL together with IoTDM is being executed or in a low cost solution near the BLE devices, for instance, a Raspberry Pi 3 Compute Module IO Board V3 with one or more Compute Modules 3 (https://www.raspberrypi.org/products/compute-module-io-board-v3/). Raspberry Pi 3 is widely used in industrial scenarios due to their robustness. In any of the two options, a BLE receptor is required (via USB or GPIO) as gateway between the BLE network and IoTDM, and a parser from BLE data to JSON and vice versa must be working in the wrapper. An open-source solution of this parser can be found in [28].

The industrial machinery networks, with devices operating with technologies and communication protocols like Modbus, CAN Bus, and Ethercat cannot operate directly with IoTDM due to their data format and communication protocols incompatibility. As we stated in the previous section, this machinery uses as intermediate step in their communications a OPC UA client-server architecture. Each technology needs an OPC UA server to convert the data to JSON format and to communicate with an OPC UA client. The OPC UA client is, at the end, the entity that must communicate with ODL and IoTDM. Note that OPC-UA server-clients are not typically open-source software. Then, two open source options are proposed in this work for enabling interoperability among industrial machinery and ODL with IoTDM:

(i) An OPC UA client-server solution for each technology, installed in an intermediate machine. This machine could be the wrapper. This must parse, adapt, and restructure the received data from OPC UA clients in order to shape a valid JSON or XML understood by IoTDM. The communication procedure of this option is plotted in Figure 4 with blue lines. In [26] a list of open source solutions can be found.

(ii) A parser for each technology which addresses separately the data conversion to XML or JSON. They could be installed in an intermediate machine (wrapper). The communication procedure of this option is plotted in Figure 4 with red lines. There are some open-source solutions that can be used: modbus TCP [29] for Modbus; ecatmod for EtherCAT [30] and libcanardbc [31] for CAN bus.

Table 3: Methods implemented in the HubPlugin class.

Method	Description
getConfig	Returns to the controller the hub configuration
getDevices	Returns to the controller the device list managed by the hub.
getDeviceConfig	Returns to the controller a concrete device configuration
getDeviceData	Returns to the controller the data given by a concrete device
getDeviceSchema	Returns to the controller the configuration scheme from a concrete device
setConfig	Updates the hub configuration
setDeviceConfig	Updates a concrete device configuration

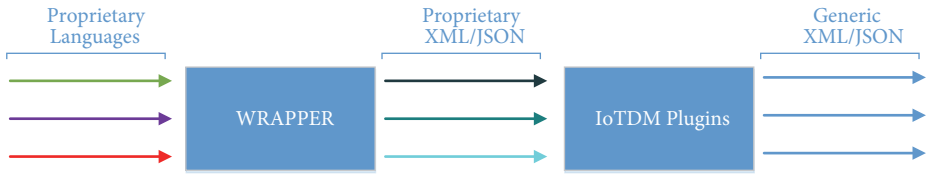

Figure 6: Data flow (upload) to reach a generic JSON/XML data format.

In any of the two options an intermediate machine is needed. In Figure 4 the wrapper is plotted as a single machine, but a set of machines could be required if the number of networks to gateway or the amount of data to process is high.

Note that the software solution proposed for industrial machinery will change in a near future because the OPC Foundation is working on a new extension to make OPC UA compatible with IoT applications, the OPC UA PubSub [32].

In short, those devices that cannot communicate directly with IoTDM make use of a wrapper to shape their data to be understood by IoTDM. The communication among the actors of this scenario is illustrated in Figure 6. The wrapper joins the functionalities of the parsers described above as a single bridge that receives the data from different technologies and parses and sends the converted data to ODL with IoTDM.

5.2. Controller Layer. ODL with IoTDM works in the controller layer. It can be run in a dedicated enterprise server or in the cloud. The latter could simplify the implementation and reduce the hardware costs. The decision about where the SDN platform must be allocated depends on the computation resources in the IT network factory, the size of the IT IoT and industrial machinery networks and the amount of data traffic to manage, the security policies in the enterprise, etc. ODL and IoTDM can be executed in Linux and Windows, but the former is better in terms of performance and compatibility. When ODLwith IoTDM is executed, it keeps awaiting incoming device petitions and/or data. The data received by IoTDM with specific JSON structure must be parsed to generic JSON structure, needed to be understood by any application in the northbound layer. Thanks to the open ODL functionality, this parser has been developed in this work. Here is where the vendor lock-in problem is completely broken.

The parser has been developed with an abstract class, *HubPlugin*, which has been defined in the controller. It includes some methods to be implemented for each new technology to adapt. This lets us isolate the communication mechanisms and the data formats of each vendor. The plugin not only provides the data in the required JSON format but also sets the communication protocol with the physical *hub* (HTTP, CoAP, or MQTT). In this software architecture a *hub* is understood as a technology involved in the physical layer (RFID, BLE, Profibus, Modbus, etc.). Each *HubPlugin* implementation must include a set of methods, summarized in Table 3.

The data in ODL with IoTDM are internally processed and stored in the Model-Driven SAL (MD-SAL) [33]. This is a set of infrastructure services defined in ODL with the aim of providing a common and generic support to applications, plugins for developers and infrastructure services for data storage, RPC, service routing and notification for publish/subscribe services. Figure 7 shows the module hierarchy of the parser implemented in the controller layer. The services are described as follows:

(i) Controller Manager is responsible for the northbound interface implementation. It also includes the controller core, adding the following tasks: (i) communication with ODL data base (MDS-SAL) and (ii) dynamic plugins instantiation (specific format to generic format translators) for enabled hubs (technologies).

(ii) Controller Provider is in charge of giving the base element executed by ODL when it starts, initiating the Controller Manager.

(iii) Data Gathering polls every configured time to all activated plugins, allowing the data storage.

(iv) ODL Data Base is the specific MD-SAL data store from ODL.

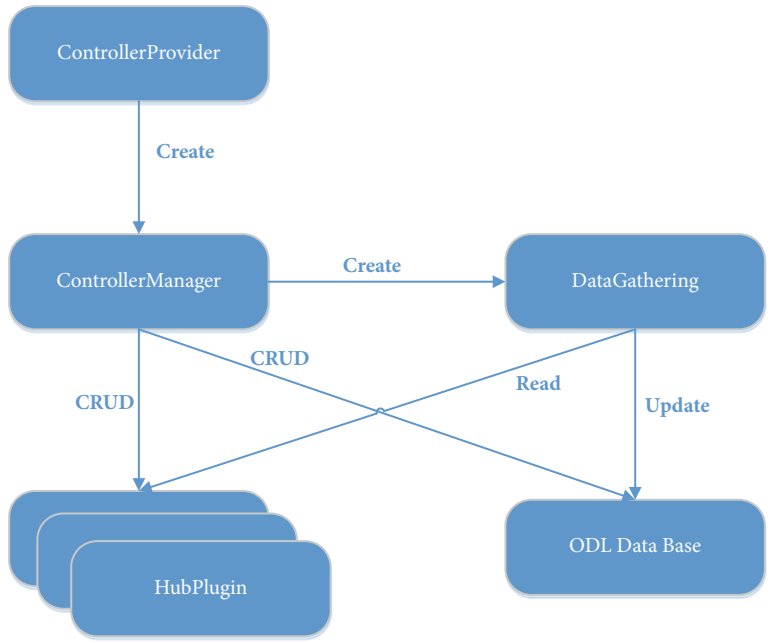

Figure 7: Modules hierarchy implemented in ODL.

(v) Hub Plugin is the abstract class which defines the behavior of each plugin and hub involved in the scenario.

5.3. Northbound Layer. The Northbound layer is the highest layer in the SDN architecture. It is composed of user applications. These integrate the so-called Northbound APIs. The main functionality of this layer is to give the users final applications with different purposes, for instance, data and device management in the network, device visualization in graphic interfaces, host tracker, flow programmer, or static routing.

Northbound applications communicate with the physical network in the Southbound through the SDN controller, which takes decisions about how to proceed. An example of Northbound API is given in Section 6, where a web application is developed with the aim of managing an IIoT network, composed of devices from different vendors. Other examples of Northbound APIs are shown in Figure 8. There is a wide variety, with different purposes: cloud orchestration tools based on OpenStack (https://www.openstack.org/), VMware (https://www.vmware.com/), or CloudStack (https://cloudstack.apache.org/) for instantiation of virtual machines in network elements; services to increase network security such as firewalls to filter traffic according to certain criteria based on ports, IP addresses, or services; network planning tools such as Net2Plan (http://net2plan.com/), to generate topologies, to collect statistics or to execute network optimization algorithms; applications of accounting and load balancing; applications focused on network operation and management, some of them based on CRUD services (Create, Read, Update, and Delete), or dashboards to show the topology and the current status of every device in the network.

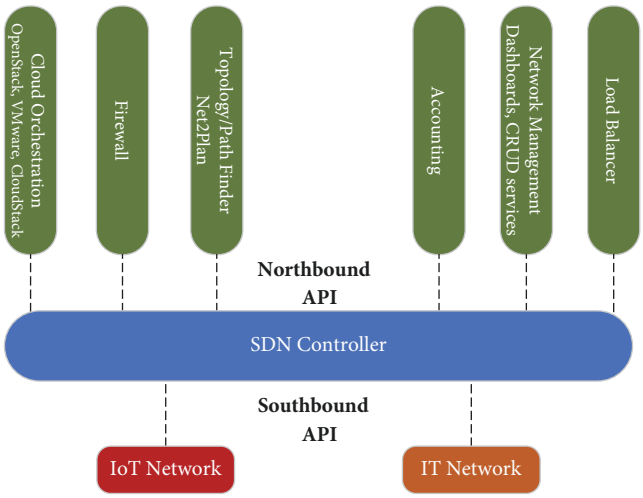

Figure 8: Northbound APIs examples.

6. IIoT Pilot Scenario Using ODLwith IoTDM and Plugins

In contrast to the works reviewed in Section 2, this work shows a pilot to test the proposed open source SDN architecture solution. This pilot provides light to the orchestration of an industrial scenario using SDN, testing the management of a set of lamps and sensors installed in an industrial warehouse, together with the IT network operating in the same factory. This scenario could be real in a medium-high factory, and it could be extended for managing any industrial technology, just following the software and hardware requisites described in Section 5.

Following the three layered architectures in Figure 1, extended in Figure 4, the southbound consists of a set of

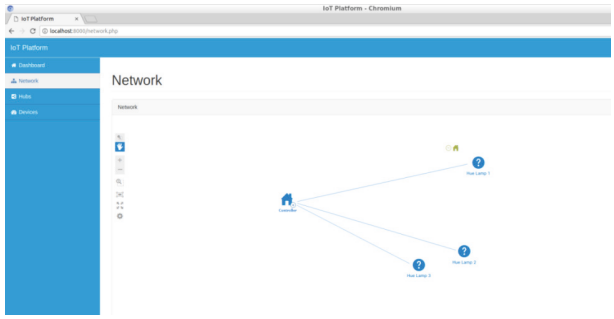

Figure 9: Dashboard of web interface for orchestrating IIoT networks.

smart lamps (Philips Hue (https://www.philips.co.uk/c-p/8718291775027/hue-personal-wireless-lighting/specifications)) and a set of intelligent sensors used in industrial scenarios for monitoring temperature, humidity, etc. (B+B SmartWorx (http://advantech-bb.com)). They work as an IIoT network. In the controller layer ODL with IoTDM, version Carbon 0.6.3 is executed in a local machine running Ubuntu- Linux. In the northbound a simple web application has been developed to test the communication among all actors in the architecture. The user-devices interactions are done in a background communication between the web interface and the controller layer. A snapshot of the web application is shown in Figure 9. It shows the visual management of those devices connected to the IIoT network and allows users to create, modify, and/or delete entities related to the deployed scenario. The web tool offers four main functionalities:

(i) Dashboard: main application screen, showing general information about the platform as the number of devices (hubs) to manage.

(ii) Network: showing the topology deployed.

(iii) Hubs: screen to manage hubs via CRUD.

(iv) Devices: screen to visualize and configure devices.

The pilot has been tested for orchestrating the single IIoT network, the IT network and both together with a single controller.

7. Conclusions

On the one hand, this work has identified the main problems that could arise in the I4.0 adoption for a medium-high factory: the management of thousands of IoT devices of different technologies and vendors, the heterogeneity of communication protocols and data formats, the need of manage the traditional industrial machinery networks and IT networks together with new IoT infrastructures, and the risks of huge amount of data traffic in the I4.0 infrastructure, which could derive in undesirable packet delays and network congestion. After that, an open-source software solution architecture based on ODL together with IoTDM has been proposed to orchestrate the whole I4.0 infrastructure, enabling interoperability and management of IIoT devices from different vendors, including conventional industrial machinery and IT networks. The software architecture solution includes the use of different system of plugins in the southbound and in the controller layer. In the latter, the plugin is based on oneM2M standard that parse the JSON/XML format from IoTDM to a generic JSON/XML format understood by any application in the northbound. The system of plugins has been developed with modularity and scalability premises, giving an open software solution that allows anyone to be able to develop a new plugin for each technology involved in an industrial scenario. The software architecture presented in this work provides scalability and exportation to other industrial scenarios, breaking the vendor lock-in barriers and generating a vendor-agnostic solution.

This work gives light to the practical difficulties appearing in introducing SDN in industrial contexts for I4.0 adoption. It reviews and provides solution to critical aspects from the controller architectural choices, specific IoT interfaces, and the difficulties for covering different communication protocols popular in industrial contexts.

Data Availability

The paper proposes an open-source software architecture solution for "orchestrating" and Industrial IoT scenario. All software components are free (available) in the references the authors have included throughout the paper. Only two software components are not open-source (available) because they are still working on them. But the authors give all details about their structure: Section 5: controller layer: middleware parser and Section 6: northbound application. The authors are planning to upload the source code in the github repository of their research group: https://github.com/girtel/.

Disclosure

This publication reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work has been partially cofunded by the Spanish National Project ONOFRE-2 (ref. no. TEC2017-84423-C3-1-P), H2020 EU Project (ref. no. H2020-ICT-2016-2 (ID 761727) topic: ICT-7-2016), and Erasmus+ Program of EU (ref. no. 575853-EPP-1-2016-1-ES-EPPKA2-SSA).

References

[1] P. Gerbert, M. Lorenz, M. Rüßmann et al., *Industry 4.0: The Future of Productivity and Growth in Manufacturing Industries*, Boston Consulting Group, 2015.

[2] L. Atzori, A. Iera, and G. Morabito, "The internet of things: a survey," *Computer Networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[3] S. Jeschke, C. Brecher, T. Meisen, D. Özdemir, and T. Eschert, "Industrial Internet of Things and Cyber Manufacturing Systems," in *Industrial Internet of Things*, Springer Series in Wireless Technology, pp. 3–19, Springer International Publishing, Cham, 2017.

[4] P. P. Ray, "A survey of IoT cloud platforms," *Future Computing and Informatics Journal*, vol. 1, no. 1-2, pp. 35–46, 2016.

[5] "5H2020 – UNIFY-IoT Project, Report on IoT platform activities," http://www.unify-iot.eu/wp-content/uploads/2016/10/D03_01_WP02_H2020_UNIFY-IoT_Final.pdf.

[6] R. Yang, Z. Wen, J. Xu, and M. Rovatsos, "Fog Orchestration for IoT Services: Issues, Challenges and Directions," *IEEE Internet Computing*, vol. 21, no. 2, pp. 16–24, 2016.

[7] D. Wilusz and J. Rykowski, "Orchestration of distributed hetero-geneous sensor networks and Internet of Things," *Informatyka Ekonomiczna*, vol. 3, no. 33, 2014.

[8] N. Matsuda, K. Takagi, S. Horiuchi, H. Aoki, and A. Akutagawa, "IoT network implemented with NFV," *NEC Technical Journal*, vol. 10, no. 3, pp. 35–40, 2016.

[9] N. Omnes, M. Bouillon, G. Fromentoux, and O. Le Grand, "A programmable and virtualized network & IT infrastructure for the internet of things: How can NFV & SDN help for facing the upcoming challenges," in *Proceedings of the 2015 18th International Conference on Intelligence in Next Generation Networks, ICIN 2015*, pp. 64–69, February 2015.

[10] N. Bizanis and F. A. Kuipers, "SDN and virtualization solutions for the internet of things: a survey," *IEEE Access*, vol. 4, pp. 5591–5606, 2016.

[11] A. Agarwal, G. Misra, and K. Agarwal, "The 5th Generation Mobile Wireless Networks- Key Concepts, Network Architecture and Challenges," *American Journal of Electrical and Electronic Engineering*, vol. 3, no. 2, pp. 22–28, 2015.

[12] Y. Li, X. Su, J. Riekki, T. Kanter, and R. Rahmani, "A SDN-based architecture for horizontal Internet of Things services," in *Proceedings of the ICC 2016 - 2016 IEEE International Conference on Communications*, pp. 1–7, Kuala Lumpur, Malaysia, May 2016.

[13] R. Vilalta, A. Mayoral, D. Pubill et al., "End-to-end SDN orchestration of IoT services using an SDN/NFV-enabled edge node," in *Proceedings of the 2016 Optical Fiber Communications Conference and Exhibition, OFC 2016*, USA, March 2016.

[14] R. Vilalta, A. Mayoral, R. Casellas, R. Martinez, and R. Muñoz, "SDN/NFV Orchestration of Multi-technology and Multi-domain Networks in Cloud/Fog Architectures for 5G Services, in Proc. of OptoElectronics and Communications Conference (OECC," in *Proceedings of the OptoElectronics and Communications Conference (OECC)*, Niigata, Japan, 2016.

[15] Y. Ma, Y. Chen, and J. Chen, "SDN-enabled network virtualization for industry 4.0 based on IoTs and cloud computing," in *Proceedings of the 2017 19th International Conference on Advanced Communication Technology (ICACT)*, pp. 199–202, Pyeongchang, Kwangwoon Do, South Korea, Feburary 2017.

[16] K. Sood, S. Yu, and Y. Xiang, "Software-defined wireless networking opportunities and challenges for internet-of-things: a review," *IEEE Internet of Things Journal*, vol. 3, no. 4, pp. 453–463, 2015.

[17] 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.

[18] X. Li, D. Li, J. Wan, C. Liu, and M. Imran, "Adaptive Transmission Optimization in SDN-Based Industrial Internet of Things With Edge Computing," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 1351–1360, 2018.

[19] W. Cerroni, C. Buratti, S. Cerboni et al., "Intent-based management and orchestration of heterogeneous openflow/IoT SDN domains," in *Proceedings of the 2017 IEEE Conference on Network Softwarization (NetSoft)*, pp. 1–9, Bologna, Italy, July 2017.

[20] J. wan, S. Tang, Z. Shu et al., "Software-Defined Industrial Internet of Things in the Context of Industry 4.0," *Proceedings of IEEE Sensors Journal*, vol. 16, no. 20, 2016.

[21] A. L. Stancu, S. Halunga, A. Vulpe, G. Suciu, O. Fratu, and E. C. Popovici, "A comparison between several Software Defined Networking controllers," in *Proceedings of the 12th International Conference on Telecommunications in Modern Satellite, Cable and Broadcasting Services, TELSIKS 2015*, pp. 223–226, Serbia, October 2015.

[22] R. Khondoker, A. Zaalouk, R. Marx, and K. Bayarou, "Feature-based comparison and selection of Software Defined Networking (SDN) controllers," in *Proceedings of the World Congress on Computer Applications and Information Systems (WCCAIS '14)*, January 2014.

[23] S. Rowshanrad, V. Abdi, and M. Keshtgari, "Performance evaluation of sdn controllers: Floodlight and OpenDaylight," *IIUM Engineering Journal*, vol. 17, no. 2, pp. 47–57, 2016.

[24] "EPCglobal, EPC Radio-Frequency Identify Protocols Class-1 Generation-2 UHF RFID Protocol for Communications at 860 MHz - 960 MHz," https://www.gs1.org/sites/default/files/docs/epc/uhfc1g2_1_2_0-standard-20080511.pdf.

[25] *Control Area Network standard (road vehicles)*, 2018, https://www.iso.org/standard/63648.html.

[26] "Open source OPC UA servers and clients," https://github.com/open62541/open62541/wiki/List-of-Open-Source-OPC-UA-Implementations.

[27] A. Tavakoli, M. Casado, T. Koponen, and S. Shenker, "Applying NOX to the Datacenter," in *Proceedings of the 8th ACM Workshop on Hot Topics in Networks (HotNets-VIII)*, vol. 2009, New York, NY, USA.

[28] "Open source BLE to JSON parser," https://github.com/mfgCode/ble-packet-parser.

[29] "Open source modbus-tcp parser," https://www.npmjs.com/package/modbus-tcp.

[30] "Open source ecatmod parser," https://www.npmjs.com/package/ecatmod.

[31] "Open source libcanardbc parser," https://github.com/Polyconseil/libcanardbc.

[32] "OPC Fundation Publish-Subscribe release," https://opcfoundation.org/news/press-releases/opc-foundation-announces-opc-ua-pubsub-release-important-extension-opc-ua-communication-platform/.

[33] "ODL controller: Model Driven SAL," https://wiki.opendaylight.org/view/OpenDaylight_Controller.