Enabling Industrial IoT as a Service with Multi-Access Edge Computing

Enabling Industrial IoT as a Service with Multi-access Edge Computing

Davide Borsatti, Gianluca Davoli, Walter Cerroni and Carla Raffaelli

***Abstract*—Industrial IoT coupled with emerging cloud computing architectures shows high potential in transforming the way industrial processes are managed and carried out. This potential can be further enhanced by enabling on-demand deployment of IoT services located very close to the factory premises. This paper proposes an architecture, based on the ETSI Multi-access Edge Computing (MEC) framework, for the automated deployment of Industrial IoT applications "as a service," taking advantage of proximity computing platforms such as edge and fog environments. A proof-of-concept implementation is reported to demonstrate that transforming Industrial IoT applications into MEC-based services running over multiple technological domains is not only feasible, but can achieve full service deployment in a matter of a few seconds.**

I. Introduction

The Internet of Things (IoT) concept has evolved in the last decade, and has expanded in many fields of today's society. An increasing number of smart environments are being created, unlocking unprecedented potentialities for innovative applications and developments [1]. One of the context attracting increasing interest for extensive IoT applications and adoption of advanced communication network technology is represented by smart industry and manufacturing, often referred to as Industry 4.0 [2]. Smart connectivity capabilities of a large number of sensors and devices, the availability of cloud computing platforms, and the implementation of software-defined network control and management techniques bring the opportunity to support rapid material handling, efficient information sharing, fault detection time reduction, and flexible production processes in manufacturing environments.

Recently, 5G technologies have been making their way inside factories as well, enabling the capability to manage a high density of devices and different classes of service, including massive machine-type communications, enhanced mobile broadband, and ultra-reliable low latency communications [3]. This is expected to make the integration of components easier, by improving the communication between heterogeneous devices. Also, Cyber-physical Systems (CPS), such as machine digital twins, which combine statistics, computer modeling, and real-time data measured on physical systems, can help in modeling the response of a system under multiple working scenarios. Industrial systems can benefit from the combination of multiple technologies and agents, in order to take real-time decisions and reach the common goal of improving the efficiency and responsiveness of production systems [4].

As an Internet-based commodity to share computing, storage, and network resources, cloud computing can be part of the answer to the increasing need of manufacturers for data processing. New cloud-based manufacturing models can be defined, where all resources and capabilities are virtualized and offered "as a service" allocated on-demand through the cloud, leading to the concept of smart manufacturing [5]. However, exchanging data between machines/sensors and remote cloud locations may result in delayed responses, high usage of bandwidth, and energy consumption. Besides, the long-distance communication exposes the system to the risk of external network faults and security breaches, which represent highly critical challenges.

To overcome those problems, fog and edge computing solutions have been proposed to bring compute, storage, and network capabilities closer to or even within the user premises. As a consequence, data collected from smart machines and sensors can be processed locally or at the edge, without reaching the cloud, to fulfill stringent requirements on latency, real-time responsiveness, limited network traffic, and protection of sensitive data. In particular, considering the nature of devices and equipment used in a factory, peripheral processing in support of highly reactive systems could be more effective than a typical cloud-based approach [6].

All the discussed aspects should be considered in order to reach the original objective of Industry 4.0 of achieving much higher gains in operational efficiency with respect to the marginal improvements expected from traditional cost-cutting measures. Therefore, it is clear that the transition to Industry 4.0 will depend on the successful adoption of many new information and communications technologies, which, in addition to enhanced IoT connectivity and processing, will provide highly flexible control and management capabilities, as well as high reliability and security. All of this will be offered to the customers in a fully automated and collaborative environment through suitable programmable platforms, thus fostering the introduction of an *Industrial IoT as a Service* (IIoTaaS) model.

Despite the availability of technologies at an adequate level of maturity, there still exist the need for a suitable framework in which the different communication and software technologies can operate efficiently to achieve the objectives of smart manufacturing and IIoTaaS. The main contribution of this paper is the definition of an architecture for IIoTaaS applications which takes advantage of a multi-level computing platform, consisting of edge, fog and cloud environments. In particular, the proposed approach aims at unifying the orchestration of heterogeneous fog and edge computing resources under a single framework, which is designed to be compliant with existing standards for Multi-access Edge Computing (MEC) [7], rather than defining a new set of interfaces. This

The authors are with the Department of Electrical, Electronic and Information Engineering "G. Marconi," University of Bologna, Italy.

brings all the advantages of MEC-based service management to the development and deployment of IIoTaaS applications. In this paper, a reference operational architecture, the different components of the framework, and a proof-of-concept implementation are reported, showing how the MEC-based approach and the supporting information and communication technologies enable the automated deployment of IIoTaaS applications in a matter of seconds. Smart industry environment is considered here, representing one of the most challenging and demanding application of the proposed framework. In any case the proposed methodology has a high potential to be reused in other fields as well.

II. Reference Scenario for IIoT as a Service

In a smart manufacturing environment, production line appliances are equipped with sensor nodes that generate and exchange monitoring data over a (wireless) network, according to IoT principles [3]. Such data then needs to be processed, to evaluate production performance and recognize faults in the procedure, as well as for other purposes. To this aim, several diverse compute and network resources are available, ranging from nearer and less powerful ones located at the edge or in one or multiple fog clusters, to farther but more powerful ones located in a remote cloud, as depicted in Fig. 1.

According to its original definition, the fog domain includes compute resources offered by infrastructures located anywhere from the cloud to the edge. For the purpose of this work, however, only fog resources located in the edge domain are considered. Therefore, in the following, the term *fog resources* identifies computing resources located in the local network of the factory, including mobile devices being carried around the premises by personnel as well as devices located on production line machinery. On the other hand, the term *edge resources* identifies local datacenters, closer to the access network or within factory premises, and possibly including high-performance servers normally employed to perform specific low-latency tasks.

Considering that IIoTaaS applications can include software components that must take advantage of the proximity to the source of data, both edge and fog resources should be used to deploy them, based on their availability. Orchestrating edge and fog resources in a unified way according to the ETSI MEC framework [7] allows to benefit from its additional features. The MEC approach can be used to facilitate the interoperability between services hosted on different domains and implemented with different technologies, towards a seamless integration of heterogeneous service components. It also allows for a direct interaction with the 5G access network, including direct knowledge of end-system position and connection quality, for which standard APIs already exist (e.g., MEC 013 for Location API, or MEC 012 for Radio Network Information API), thus enabling the support of new types of services. This reference scenario still supports a more classic cloud-based approach, in which the computational resources are located in remote datacenters, public or private, offering higher capacity than the one provided by local resources.

Recent work includes valuable examples of the adoption of fog and edge paradigms in 5G and Industrial IoT contexts.

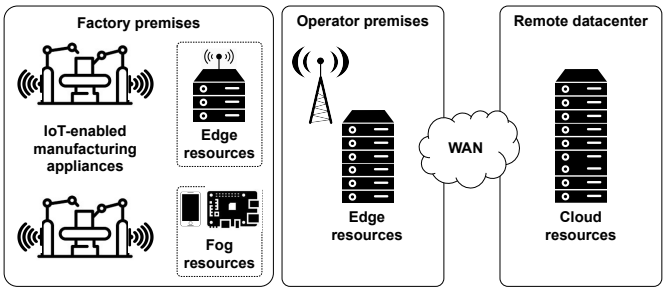

Fig. 1. Reference scenario, where cloud, edge and fog computing resources are available; in specific cases, only fog or edge resources may be available in addition to cloud ones.

A number of such scenarios are presented in [8], but no specific management architecture dealing with the diversified infrastructure is proposed. On this matter, some architectures integrate NFV and Fog [9], but do not adopt standardized functionalities that can be provided at the edge. Other research efforts target a scenario that is very similar to the one presented here [10] [11], but consider a fog infrastructure including fixed hardware only. In contrast, this paper proposes a management architecture based on ETSI MEC and ETSI NFV standards to supervise remote cloud, edge and fog resources, with the benefits that come with the integration of a MEC system, including service discovery, location services, and more. Also, in this work fog clusters comprise devices that are not necessarily known a-priori, considering the possibility of allocating services on them in a dynamic and automated fashion. Finally, the fog domain service orchestrator employed here is able to allocate the service according to multiple allocation models, and to choose the most efficient allocation technique available at the time of service request [12].

In line with one of the use cases presented in [8], a possible example of an IIoT service spanning across multiple domains could be based on data exchanged according to a publish-subscribe paradigm. A set of remote cloud resources could be allocated to store and analyze long-term data received from the factory premises. A set of edge resources may be allocated to act as brokers between the sources of IoT data and their subscribers, being them running in the local or remote domain. Furthermore, local processing of the aforementioned data may be needed to comply with the requirements of low latency services. Finally, exploiting the proximity and mobility of fog devices, their resources may be used on demand to collect data from specific areas and/or appliances of the factory, to perform light processing on them, and to redirect pre-processed information toward the message broker and, in turn, to interested subscribers when needed. All the data exchanges just described could use typical IoT messaging protocols, whose components must then be dynamically deployed according to the specific service needs.

III. Features and Components in a MEC-enabled IIoTaaS Framework

A framework and reference architecture for MEC is introduced in [7], along with the description of relevant functional elements. Based on this reference architecture and the scenario

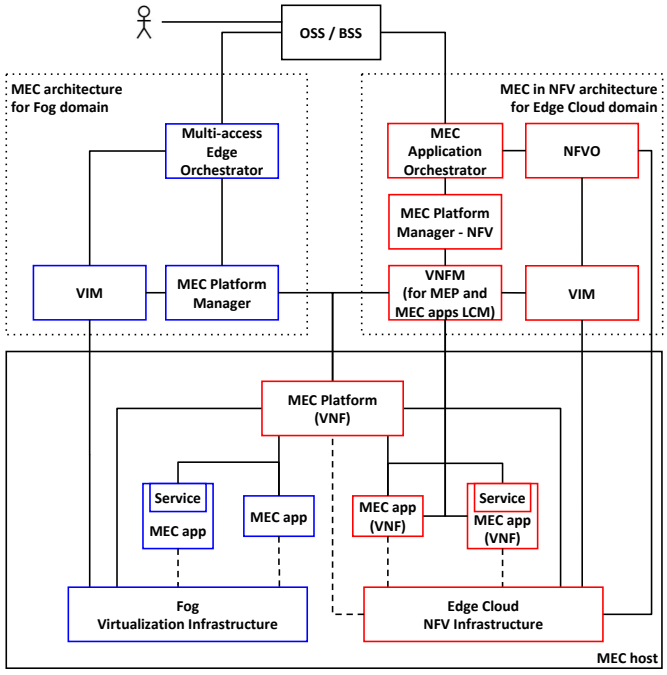

Fig. 2. MEC-compliant proposed architecture, incorporating elements from the edge (highlighted in red) and fog (highlighted in blue) computing environments.

described in Section II, the proposed MEC-compliant architecture for IIoTaaS application deployment and resource allocation across edge and fog computing environments is shown in Fig. 2. The supervising entity is the Operation/Business Support System (OSS/BSS), representing users or third-party services that manage service deployment, enforce company policies, or react to computational needs.

Service deployment in the fog computing subsystem is managed by FORCH [12] (Fog ORCHestrator, or FO for brevity), a modular orchestrator for flexible deployment of computing services over dynamic fog computing infrastructures. The components of FORCH can be mapped onto functional blocks described in the reference MEC standard. With reference to the left side of Fig. 2, this mapping is described as follows:

- the functionalities of the Multi-access Edge Orchestrator are realized by the FO mediator module, which receives service requests and selects the appropriate resources to be allocated to the service;
- tasks pertinent to the MEC Platform Manager are executed by the FO aggregator module, which gathers and aggregates information on service deployment and resource usage;
- the Virtualization Infrastructure Manager (VIM) defined in the MEC standard is mapped onto the FO VIM module, which manages the activation of services on fog resources, handling technology-specific details required by allocation and deployment procedures, and collecting monitoring information;
- the Fog Virtualization Infrastructure (VI) is represented by the VI offered by each of the fog nodes, which hosts activated services running on local resources, and reports monitoring information to the FO VIM module.

The latter component is meant to be configured on each of the nodes offering resources to the fog system. The resource utilization of every node is monitored, and this information is passed to the fog orchestrator, along with the set of services that the node can host, allowing the orchestrator to make informed decisions on the allocation of services on nodes. Allocation policies are configurable to meet specific needs, via a multi-tenant system, based on an approach similar to that of major cloud orchestrators.

Considering the right side of Fig. 2, the architecture proposed for the edge domain is essentially based on the "MEC in NFV" architecture defined by ETSI in [7], which aims at reusing components from the Network Function Virtualization (NFV) framework to fulfill a part of the MEC management and orchestration tasks, thus allowing to instantiate both MEC applications and Virtualized Network Functions (VNFs) within a unified framework. Specifically, the functionalities offered by the Multi-access Edge Orchestrator (MEO) are split into two different functional blocks, the NFV Orchestrator (NFVO) and the MEC Application Orchestrator (MEAO). The former is in charge of managing MEC applications as if they were typical VNF instances. The latter is in charge of the remaining MEO functions, such as enabling the instantiation and termination of MEC applications and maintaining a view of the status of the MEC system (e.g., deployed MEC hosts, available resources). The MEC Platform Manager becomes the MEC Platform Manager - NFV (MEPM-V), which is in charge of the same tasks as the MEC Platform Manager but without any Life-Cycle Management (LCM) action on the MEC Platform. These actions are instead delegated to a Virtualized Network Function Manager (VNFM), being the MEC Platform itself deployed as a VNF.

The edge and fog computing environments are merged at the MEC host level. A MEC host functional block can be implemented as a single physical or virtual machine offering computing resources, or as a cluster of such machines, operating behind a single abstraction. Therefore, the set of fog nodes and edge resources can be grouped into a single MEC host, operating with a single MEC Platform (MEP). The choice of considering both the edge and fog domains as a single MEC host is motivated by the particular characteristics of fog nodes. Having assumed the set of fog resources to be mutable over time, instantiating and maintaining an active MEC Platform in this kind of scenario could be challenging. For this reason, the proposed architecture relies on a single MEC Platform for both domains, hosted in the edge, considering it to be a more stable infrastructure over time. This solution also enables the adoption of a single abstraction for heterogeneous computing environments (fog and edge) and simplifies the interoperability between the two domains since it does not require to handle the communication between different MEC Platforms.

IV. Proof-of-Concept Implementation

In order to demonstrate the feasibility of the proposed MEC-based architecture for IIoTaaS applications and how it enables interoperability between the edge and fog domains, a

proof-of-concept (PoC) implementation[1] has been developed and tested on commercial off-the-shelf servers under a use case relevant to Industry 4.0 scenarios. The example briefly discussed in Section II is considered, where the widely adopted Message Queuing Telemetry Transport (MQTT) protocol has been chosen as a publish-subscribe solution: an MQTT broker is deployed on edge resources, multiple sensing applications are instantiated on available fog nodes and edge nodes, acting as MQTT publishers sending data to the MQTT broker, and a sink application running in the cloud acts as an MQTT subscriber receiving data from the MQTT broker. All the required software components running in the edge and fog environments are instantiated on demand using the automated procedures offered by the proposed framework architecture and detailed below. By taking advantage of the MEC-based approach, those software components are deployed and made capable of interacting with each other independently of the specific computing platform being used, resulting in an Industrial IoT application truly offered "as a service."

The interoperability among the different computing domains introduced in Section II is achieved by making use of a testbed composed of several platforms:

- OpenStack is employed in the core cloud domain to instantiate Virtual Machines (VMs). For the PoC no specific configuration was needed.
- Kubernetes orchestrates the deployment of containers that execute MEC applications in the edge domain. For this PoC, the Kubernetes cluster was configured with Docker as a container engine, CoreDNS as DNS service, Calico as container networking solution, Metallb as load balancer, and OpenEBS as persistent volume manager.
- Open Source MANO (OSM) handles the deployment of NFV Management and Orchestration services in the edge. Specifically, it is used to deploy MEC applications in the Kubernetes cluster.
- FORCH takes care of the deployment of containers that execute MEC applications in the fog cluster.

As mentioned in Section III, the fog orchestrator FORCH is a Python-based original solution for fog computing service deployment recently developed at the University of Bologna [12], composed of several cooperating modules. In this PoC, a subset of the APIs offered by the fog orchestrator is utilized to deploy MEC services in the fog domain.

The MEC applications employed to test this solution are based on the "Unibo MEC API Tester," also developed at the University of Bologna as part of the ETSI NFV&MEC Plugtests 2020 [13]. It already implements most of the MEC 011 APIs [14], which were integrated with a simple MQTT client service that can be exposed and consumed through the aforementioned APIs. Due to the limited scope of this PoC and constraints in the experimental platform, this setup only employs the MEC 011 APIs for service registration/discovery. However the same architecture is also capable of supporting different MEC-enabled services.

To the best of our knowledge, a working open-source software tool that implements all the functionalities of a MEC Platform is not available yet. Therefore, a Python-based custom solution has been developed to implement the set of MEC Platform features required by the PoC. This custom solution follows the directive defined by ETSI in terms of API specification.

The MEC 011 APIs are adopted to aid the interaction between different MEC applications and their services, even if they are deployed on different infrastructures. More specifically, each MEC application can register its own services to the MEC Platform through a REST API POST request to the `/applications/{appInstanceId}/services` endpoint. This request includes all the details of the specific service being registered, such as hostname or IP address, transport layer protocol, and port, and/or other endpoint information. The MEC Platform receives this request and loads it into its internal database. The list of registered services can be retrieved by any other MEC application via the `/services` REST endpoint of the MEC Platform. This way, MEC applications can discover the list of available services and how to consume them. An example of the format standardized by ETSI used to describe MEC services is included below.

As specified in the ETSI MEC standard, the MEC Platform should also provide DNS resolution to all MEC applications in its domain. In the presented scenario, this would ease the communication between services deployed on the edge and the fog computing domains. It would also partly justify the choice of considering both edge and fog nodes as parts of a single MEC host abstraction. The internal DNS service of Kubernetes was configured to be exposed outside of the Kubernetes cluster, to be used also by MEC applications deployed in the fog domain. Then, two different DNS zones were defined, one for services running in the edge domain and another for those running in the fog domain. For the former, CoreDNS was configured to resolve all incoming requests related to the `mec.host` zone to external IP addresses used by the Kubernetes cluster, thus reachable by any other MEC application running in either edge or fog nodes. As for the DNS zone related to services running in the fog, the fog orchestrator adds its DNS associations to a DNS zone file that is shared with the Kubernetes DNS service, which refers to it for all the requests directed to the `fog.host` zone.

Having deployed all the necessary components, the practicability of the proposed solution is verified through an experiment that follows these steps:

1) Deployment of the MEC Platform as a Kubernetes application in the edge.
2) Instantiation of the data sink as a VM in the core cloud, which may happen before, during, or after the execution of the previous step.
3) Deployment via OSM of an MQTT broker as a MEC application running in the edge. The MQTT broker registers its service to the MEC Platform, providing details of the exposed MQTT endpoint according to a standardized JSON format, as reported in the following extract:

```
{"serInstanceId": "Mec-Broker",
 ...
```

[1]https://github.com/DavideBorsatti/IIoTaaS

```
"transportInfo":{
 ...
 "type": "MB_TOPIC_BASED",
 "protocol": "MQTT",
 "endpoint":{
  "addresses":[{
   "host" : "mec-broker.mec.host",
   "port" : "1883"}]
  } } }
```

Specifically, the `transportInfo` section contains information such as the `type` of messaging mechanism used (e.g., a topic-based message bus which routes messages to receivers based on topic subscriptions), the `protocol` used (MQTT in the example) and on which `endpoint` the service is available.

4) Subscription by the sink in the core cloud to the MQTT broker application.
5) Deployment via the fog orchestrator of an MQTT publisher as a MEC application running in the fog domain. The application `Mec-app` registers its service to the MEC Platform by simply exposing the REST endpoints to be used to start or stop the generation of MQTT traffic, therefore the value of the key `endpoint` will change to `uris`, which is a list of two elements `mec-app.fog.host/start-sensing` and `mec-app.fog.host/stop-sensing` respectively. Of course in this case the `type` and `protocol` fields in the service descriptor will be different, with `"type": "REST_HTTP"` and `"protocol": "HTTP"`.
6) Generation of MQTT traffic from the MEC application in the fog node, initiated by a REST call to its `/start-sensing` endpoint.
7) Deployment via OSM and Kubernetes of another MQTT publisher as a MEC application running in the edge domain. The application registers its service (start/stop sensing) to the MEC Platform.
8) Generation of MQTT traffic from the MEC application in the edge node.
9) Interruption of MQTT traffic generated by the MEC application in the fog domain, caused by a REST call to its `/stop-sensing` endpoint.
10) Interruption of MQTT traffic generated by the MEC application in the edge domain.

The described steps are represented in the sequence diagram of Fig. 3, limited to steps 3) to 6) for readability reasons. The deployment of any new MEC application in the fog or edge domain will follow the same steps as for the deployment of the first MEC application in the fog or the MQTT broker in the edge domain, respectively. In the experiment, multiple MQTT publishers were deployed in the form of additional MEC applications, following the steps described above.

In this PoC, MQTT was the only protocol employed to transmit sensor data. However, the system is completely agnostic to the protocol of choice. For example, OPC UA, a common industrial protocol, still uses TCP or HTTP/S as underlying transport, therefore its MEC 011 service definition would still be similar to the one described for MQTT, with `"protocol": "TCP/HTTP"` and of course with the correct host and port pair.

V. Evaluation

The proposed architecture implementation is evaluated by measuring the response time of the most relevant APIs and the footprint of the required running components. The interoperability of different computing platforms is validated by observing the MQTT traffic generated by MEC applications in different domains. Along with the partial representation of the PoC steps, Fig. 3 reports the average values of the measured time required for the system to perform each individual action. This assessment shows that the time needed to deploy an MQTT MEC application and start the MQTT message flow towards a broker heavily depends on the domain chosen, ranging from slightly more than 1s for deployments in the fog domain, to almost 16s in the edge domain. It is however worth to mention that the total time required to deploy from scratch the first complete working MQTT mechanism for this PoC, including the MQTT broker in the edge and an MQTT publisher in the fog, is marginally higher than 17 seconds, proving the advantage offered by the proposed MEC-based architecture to automate the deployment of IIoTaaS applications.

A foreseeable bottleneck resides in the MEC Platform itself. As shown in Fig 3, MEC applications need to interact with it to register and discover services. The time required to perform this operation has been observed to grow linearly with the amount of simultaneous requests and the number of registered services. However, improved performance of the MEC Platform can be achieved by applying the scaling mechanisms offered by Kubernetes.

Table I reports the amount of storage and memory resources each node needs to support hosting MEC applications, in the two considered domains. Both Kubernetes and FORCH are configured to use the same container runtime engine (i.e., Docker), but the former platform is designed for more general and complex scenarios, thus requiring more software components running in the edge nodes to operate the cluster. As expected, the results highlight that the resource utilization on fog nodes is smaller compared to edge nodes. Furthermore, the resource consumption of the employed MEC Platform is comparable to that of MEC applications. Their container images occupy about 60 MB of storage space and require approximately 21 MB of RAM to be run.

TABLE I
Resource utilization in different computing domains.

Domain	Disk util. [MB]	RAM util. [MB]
Edge	1366	202
Fog	124	180

In Fig. 4, every rising/falling edge of the curve represents the activation/deactivation of an MQTT message flow, as perceived by the only subscriber deployed, i.e., the sink application located in the cloud domain and subscribed to all MQTT topics. The line represents the amount of MQTT traffic received by said subscriber, corresponding to the sum of all

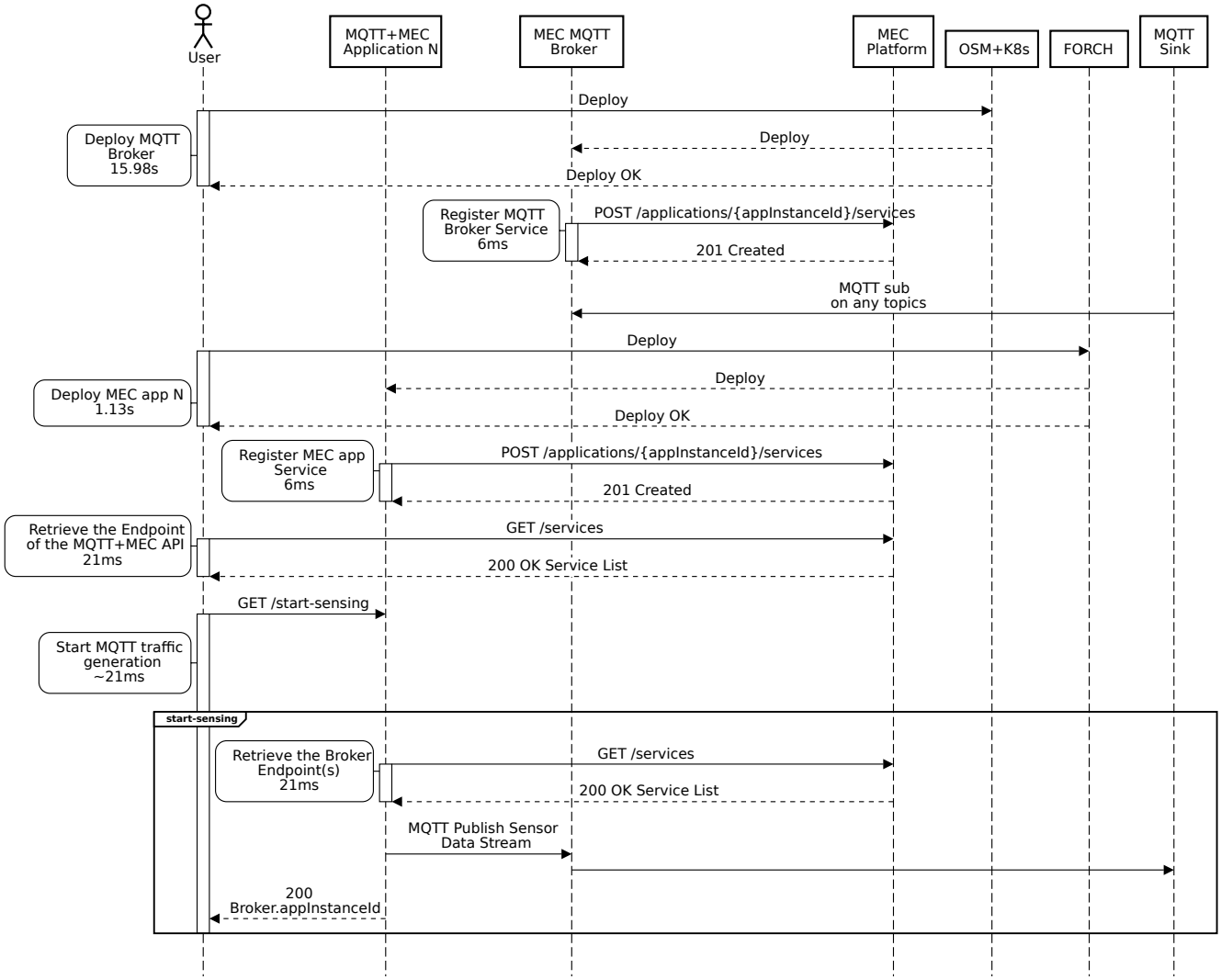

Fig. 3. Sequence diagram of part of the described PoC evaluation, with measured response times.

MQTT data flows generated by all MQTT publishers that are active at a given time. MQTT publishers are deployed as MEC applications both in the fog and in the edge domains and are activated according to an alternating pattern. Specifically, the first publisher to be activated resided in the fog domain, the second one in the edge domain, the third one in the fog, and so on. MQTT traffic generation only lasts for a limited amount of time, after which the publisher stops generating data and remains silent. In this particular example, in order to keep the figure readable, a maximum of five concurrent MQTT clients were kept active at any given time. However, this is not a generic upper bound, which would depend on available resources. The asynchronous activation of publishers causes a variable superposition of MQTT traffic at the subscriber, resulting in a step-shaped curve. The fact that the subscriber receives MQTT traffic from all publishers, regardless of the technological domain they are deployed onto, proves the effective interoperability of the proposed solution.

In this paper we focus on the application/service deployment process and limit our proof of concept to the management plane aspects of the proposed architecture. As for the data-

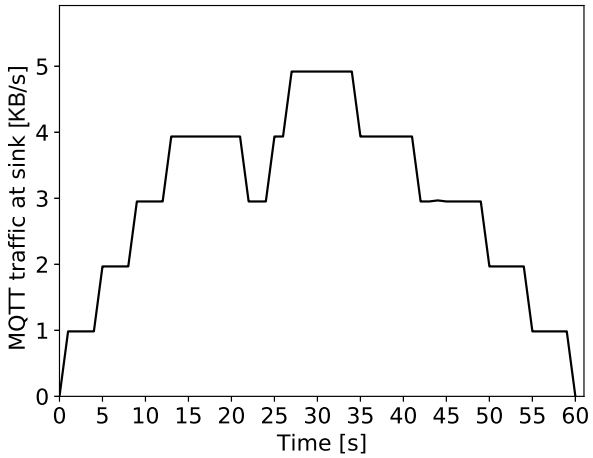

Fig. 4. Evolution of MQTT traffic received by the MQTT subscriber (sink) running in the core cloud, while varying the number of MQTT publishers and their deployment domain.

plane performance, which depends on hardware capacity, adopted technology, and geographic location, there is no general consensus on typical values for the latency in the three different domains this work considers, and only generic assumptions are typically made [15].

However, based on practical experience, it is reasonable to consider the data plane latency to be below 1ms for fog resources, around 5ms for edge resources, and around 50ms for core cloud resources.

VI. Conclusion

A novel architecture to deploy ETSI compliant Multi-access Edge Computing for Industrial IoT has been proposed and demonstrated by a proof-of-concept experiment. The solution enables inter-operation among fog, edge and cloud computing to configure and provide services with a high degree of flexibility and adaptability in a complete Industry 4.0 context. MEC applications can be seamlessly deployed over multiple technological domains in a matter of tens of seconds and can operate over different computing platforms thus transforming the Industrial IoT into a service, namely IIoTaaS. The next step will be to evolve the proposed architecture following upcoming relevant standardization activities (e.g., ETSI MEC), believing this could foster the industrial adoption of the framework as a standard-compliant solution.

References

[1] E. Ahmed, I. Yaqoob, A. Gani, M. Imran, and M. Guizani, "Internet-of-things-based smart environments: state of the art, taxonomy, and open research challenges," *IEEE Wireless Communications*, vol. 23, no. 5, pp. 10–16, 2016.

[2] T. Taleb, I. Afolabi, and M. Bagaa, "Orchestrating 5G network slices to support industrial internet and to shape next-generation smart factories," *IEEE Network*, vol. 33, no. 4, pp. 146–154, 2019.

[3] J. Cheng, W. Chen, F. Tao, and C.-L. Lin, "Industrial IoT in 5G environment towards smart manufacturing," *Journal of Industrial Information Integration*, vol. 10, pp. 10–19, 2018.

[4] Y. Cai, B. Starly, P. Cohen, and Y.-S. Lee, "Sensor data and information fusion to construct digital-twins virtual machine tools for cyber-physical manufacturing," *Procedia Manufacturing*, vol. 10, pp. 1031–1042, 2017.

[5] F. Tao, L. Zhang, Y. Liu, Y. Cheng, L. Wang, and X. Xu, "Manufacturing service management in cloud manufacturing: Overview and future research directions," *J. Manuf. Sci. Eng.*, vol. 137, no. 4, 2015.

[6] Q. Qi and F. Tao, "A smart manufacturing service system based on edge computing, fog computing, and cloud computing," *IEEE Access*, vol. 7, pp. 86 769–86 777, 2019.

[7] Multi-access edge computing (MEC); framework and reference architecture. Accessed Dec. 30, 2020. [Online]. Available: https://www.etsi.org/deliver/etsi_gs/MEC/001_099/003/02.02.01_60/gs_mec003v020201p.pdf

[8] G. S. S. Chalapathi, V. Chamola, A. Vaish, and R. Buyya, "Industrial Internet of Things (IIOT) applications of edge and fog computing: A review and future directions," *arXiv preprint arXiv:1912.00595*, 2019.

[9] P. Habibi, S. Baharlooei, M. Farhoudi, S. Kazemian, and S. Khorsandi, "Virtualized sdn-based end-to-end reference architecture for fog networking," in *2018 32nd International Conference on Advanced Information Networking and Applications Workshops (WAINA)*. IEEE, 2018, pp. 61–66.

[10] R. Vilalta, V. López, A. Giorgetti, S. Peng, V. Orsini, L. Velasco, R. Serral-Gracia, D. Morris, S. De Fina, F. Cugini *et al.*, "Telcofog: A unified flexible fog and cloud computing architecture for 5g networks," *IEEE Communications Magazine*, vol. 55, no. 8, pp. 36–43, 2017.

[11] P. Pop, B. Zarrin, M. Barzegaran, S. Schulte, S. Punnekkat, J. Ruh, and W. Steiner, "The fora fog computing platform for industrial iot," *Information Systems*, vol. 98, p. 101727, 2021.

[12] G. Davoli, D. Borsatti, D. Tarchi, and W. Cerroni, "FORCH: An orchestrator for fog computing service deployment," in *2020 IFIP Networking Conference (Networking)*. IEEE, 2020, pp. 677–678.

[13] Unibo MEC API Tester. Accessed Dec. 30, 2020. [Online]. Available: https://mecwiki.etsi.org/index.php?title=MEC_Ecosystem

[14] Multi-access edge computing (MEC); edge platform application enablement. Accessed Dec. 30, 2020. [Online]. Available: https://www.etsi.org/deliver/etsi_gs/MEC/001_099/011/02.02.01_60/gs_MEC011v020201p.pdf

[15] P. Habibi, M. Farhoudi, S. Kazemian, S. Khorsandi, and A. Leon-Garcia, "Fog computing: a comprehensive architectural survey," *IEEE Access*, vol. 8, pp. 69 105–69 133, 2020.

Davide Borsatti received his B.S. and M.S. in Telecommunications Engineering from University of Bologna in 2016 and 2018 respectively. He is currently enrolled in the Electronics, Telecommunications, and Information Technologies Engineering Ph.D. program at the University of Bologna. His research interests include NFV, SDN, Intent Based Networking, MEC, and 5G Network slicing.

Gianluca Davoli (Member, IEEE) is a Post-Doc Researcher at the University of Bologna, Italy. His research interests revolve around communication networks, focusing on the new approaches to programmability, management, and monitoring of software-based network infrastructures.

Walter Cerroni (Senior Member, IEEE) is an Associate Professor of communication networks with the University of Bologna, Italy. His most recent research interests include software-defined networking, network function virtualization, service function chaining, intent-based networking, service models for fog/edge computing platforms. He serves/served as a Series Editor for the IEEE Communications Magazine, an Associate Editor for the IEEE Communications Letters, and a Technical Program Co-Chair for IEEE-sponsored international workshops, symposia and conferences.

Carla Raffaelli (Senior Member, IEEE) is Associate Professor at the University of Bologna, Italy. She received her M.Sc. and Ph.D degrees in Electronic and Computer Engineering (University of Bologna, Italy), in 1985 and 1990, respectively. Her research interests include performance analysis and optimization of communication networks,switch architectures, optical networks and 5G networks. She regularly acts as a reviewer for top international conferences and journals. She is a member of the editorial board of the journal Photonic Network Communications by Springer and associate editor of IEEE OJ-COMS.