Baldoni, G., et al. Managing the far-Edge: are today's centralized solutions a good fit. *IEEE Consumer Electronics Magazine, 12*(3), May 2023, Pp. 51-61

DOI: 10.1109/MCE.2021.3082503

Managing the far-Edge: are today's centralized solutions a good fit?

Gabriele Baldoni
ADLINK Technology Inc, France

Luca Cominardi
ADLINK Technology Inc, France

Milan Groshev
University Carlos III of Madrid, Spain

Antonio de la Oliva
University Carlos III of Madrid, Spain

Angelo Corsaro
ADLINK Technology Inc, France

***Abstract*—Edge computing has established itself as the foundation for next-generation mobile networks, IT infrastructure, and industrial systems thanks to promised low network latency, computation offloading, and data locality. These properties empower key use-cases like Industry 4.0, Vehicular Communication and Internet of Things. Nowadays implementation of Edge computing is based on extensions to available Cloud computing software tools. While this approach accelerates adoption, it hinders the deployment of the aforementioned use-cases that requires an infrastructure largely more decentralized than Cloud data centers, notably in the far-Edge of the network. In this context, this work aims at: *(i)* analyzing the differences between Cloud and Edge infrastructures, *(ii)* analyzing the architecture adopted by the most prominent open-source Edge computing solutions, and *(iii)* evaluating those solutions in terms of scalability and service instantiation time in a medium-size far Edge system. Results show that mainstream Edge solutions require powerful centralized controllers and always-on connectivity, making them unsuitable for highly decentralized scenarios in the far-Edge where stable and high-bandwidth links are not ubiquitous.**

I. Introduction

Cloud computing has proven to be effective in managing web-based and web-scale applications but it falls short in latency sensitive applications, such as Industry 4.0 (I4.0) (e.g., robotic control) or Vehicle to everything (V2X) [1]. This limitation is due mainly to the lack of control of the connectivity between the Cloud and end-users, which spans across different Internet Service Providers (ISP). Such kind of applications requires responsiveness at a machine time-scale while today's Cloud applications mostly require human scale responsiveness.

To address this connectivity limitation, in the last years, Edge computing has arisen as a new paradigm in computing. Edge computing aims to provide compute, networking, and storage capabilities at the border of the network, closer to end-users, while providing the same *elasticity* and pricing model of Cloud computing (pay-as-you-go). This approach led to the birth of different initiatives: *(i) Multi-Access Edge Computing* (MEC) [2], from Telco industry, as a extension of Network Function Virtualization (NFV) [3] technologies closer to the access-network; *(ii) hybrid-cloud*, from IT industry, as a way to manage on-premises infrastructure from Cloud operators; and *(iii) Fog computing* [4], from manufacturing world, as an extension of Industrial IoT (IIoT).

Based on this consideration, the main contributions of this work are:

- An analysis of the overall differences between Cloud and Edge infrastructure, by diving into how they are composed and managed.
- A study of the design and deployment options of the most prominent open-source Edge solutions.
- An evaluation of the selected Edge solutions with a focus on their footprint and scalability in the presence of decentralized and relatively

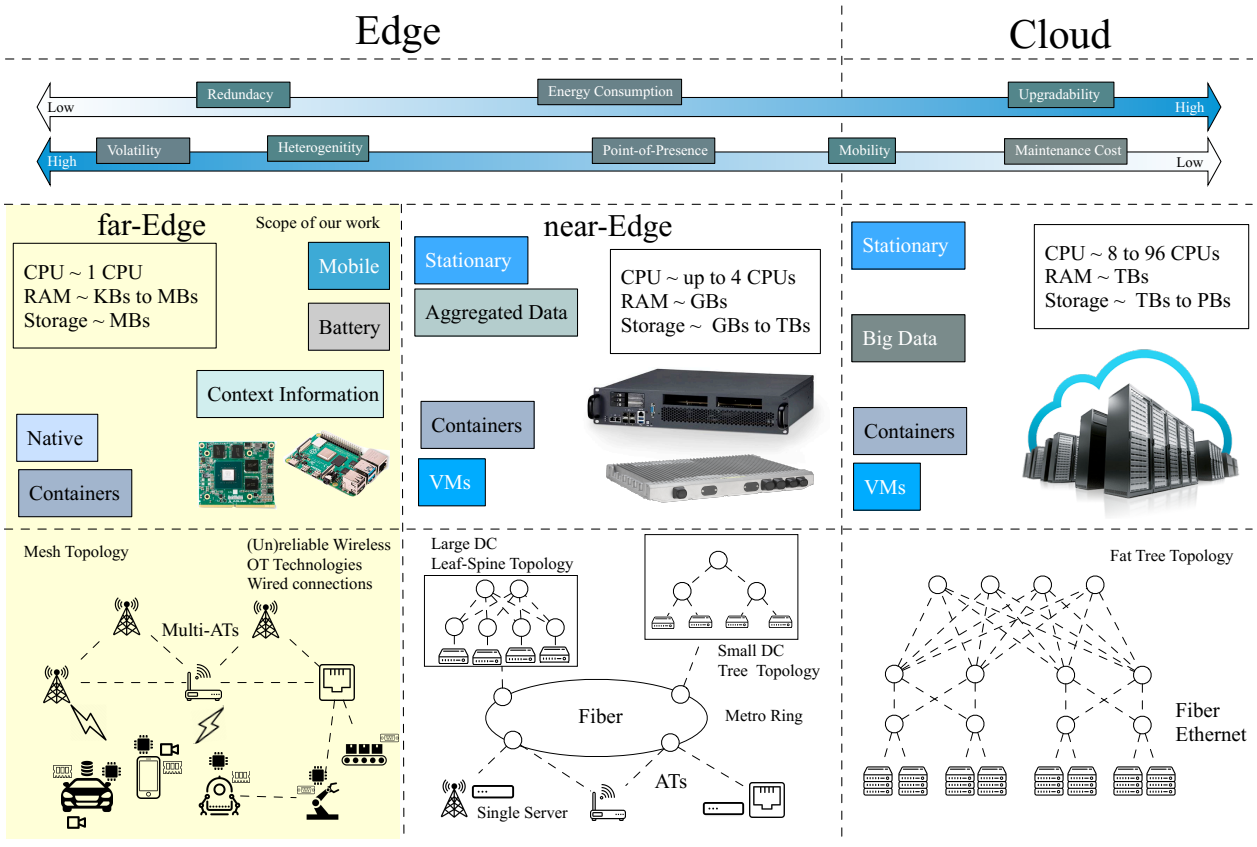


Fig. 1: Cloud vs Edge

resource constrained devices.

- A discussion on the selected Edge solutions with regards to their behavior and potential implication of their design choices.

Those initiatives refer to different *Edge concepts*: while MEC enables application developers and content providers to leverage Cloud computing capabilities and an IT service environment at the edge of the network, hybrid-cloud combines in-premises infrastructure and public-cloud allowing data and applications to be shared between them. Additionally, Fog computing distributes resources and services across Cloud, Edge, and devices on the field to create a *cloud-to-thing continuum*. Although these approaches have different scopes, their goals is the same: reduce *latency*, *jitter* and improve *Quality-of-Service*(QoS) by bringing compute, storage, and networking closer to end-users. To accommodate this trend, the Edge is therefore moving towards a more decentralized infrastructure [5] in which every device can host applications and services. This will ultimately result in a very distributed system that needs to be properly managed whilst exposing a very familiar Cloud-like model to application developers for having a chance to be successful.

II. Differences between Cloud and Edge Infrastructure

From an infrastructure perspective, Cloud and Edge domains differ under essential aspects, such as network topology, heterogeneity, and volatility as summarized in Fig. 1. Cloud computing relies on powerful and well interconnected servers that reside in a physically-secured data-center, usually located in an area where both space and energy are cheap [6], notably in remote locations far from end-users. Keeping in mind that data-centers can only be accessed via the Internet, it's no surprise that the major source of latency, jitter, and throughput limitations in Cloud application often resides in the network between the end-users and the Cloud. While those limitations are not

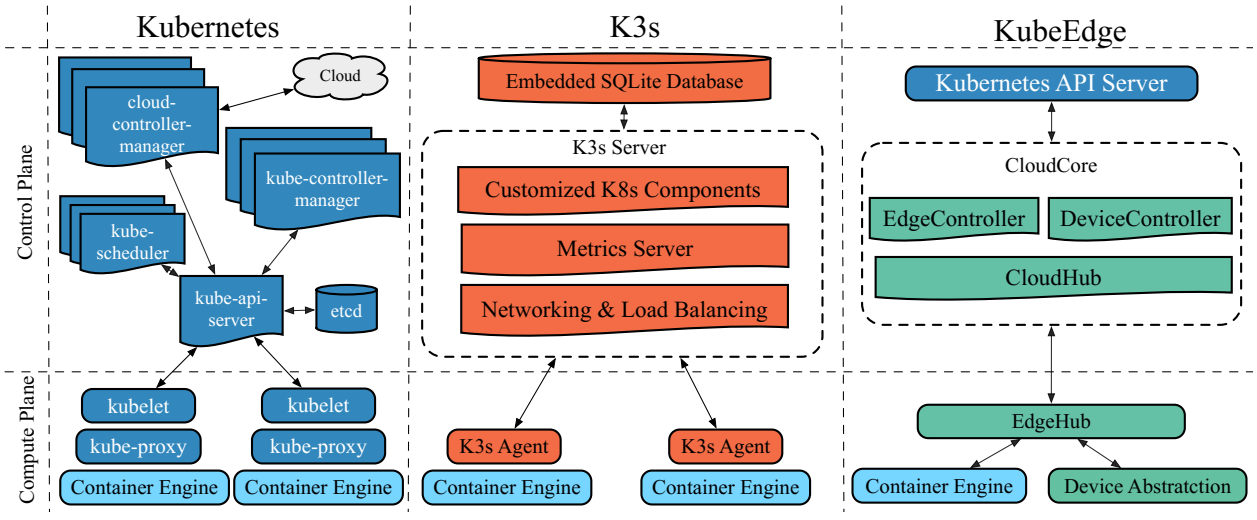


Fig. 2: Kubenetes, K3s, and KubeEdge architectures

strict for applications driven by human-scale responsiveness like the web, email, AI-training, or instant messaging, they are far to be compatible with machine-scale responsiveness required by a different class of applications like autonomous driving and robotics.

To overcome such limitations, Edge computing considers an infrastructure that goes from the edge of the network to devices that are in the field such as cars, robots or smartphones. Such infrastructure can be subdivided further into two domains as shown in Fig. 1: the near-Edge and the far-Edge. The former presents some similarities with the Cloud environment, resources are still organized in data-centers, leveraging virtualization technologies to increase application density. However, resources present heterogeneity in both computing [7] and networking [8] to cope with the geographical distribution of such small data-centers. Examples of near-edge infrastructure are given by metropolitan deployments comprising cabinets, central offices, and small data-centers. Such infrastructure allows more latency sensitive applications like CDN content caching or Cloud gaming to be deployed closer to the user, improving the overall experience. Although improving experience for human users, such infrastructure introduces a minor delay and jitter that may not still be suitable for latency sensitive applications like control-loops in robots, cars, and drones.

On the other hand, the far-Edge is composed of resources that are by nature heterogeneous, mobile and volatile. Such mobility and dynamicity of resources makes the whole infrastructure in the far-Edge to be dynamic in both time and space, making it very variegate and unstructured a priori. Resources usually leverages on Radio Access Technologies (RAT) to interact between themselves or to communicate with the near-Edge infrastructure in order to offload the resource extensive tasks like AI-training. Cooperation among these resources is expected to provide low-latency services [5], enabling use-cases requiring machine-scale responsiveness, because of time constraints coming from control-loops. Example of far-Edge resources are: cars, robots, drones or smartphones. While these resources are far less powerful compared to near-Edge and Cloud ones, they are powerful enough to host a large set of applications, such as robot intelligence, car infotainment, navigation, obstacle avoidance. Such applications can take advantages of cooperation and context information to take local decisions and quickly respond to external events without interacting with near-Edge or Cloud infrastructure.

III. Cloud and Edge Management Solutions

To better understand the challenges of managing the far-Edge, let's depart from some of the management solutions already available today. The most prominent Cloud

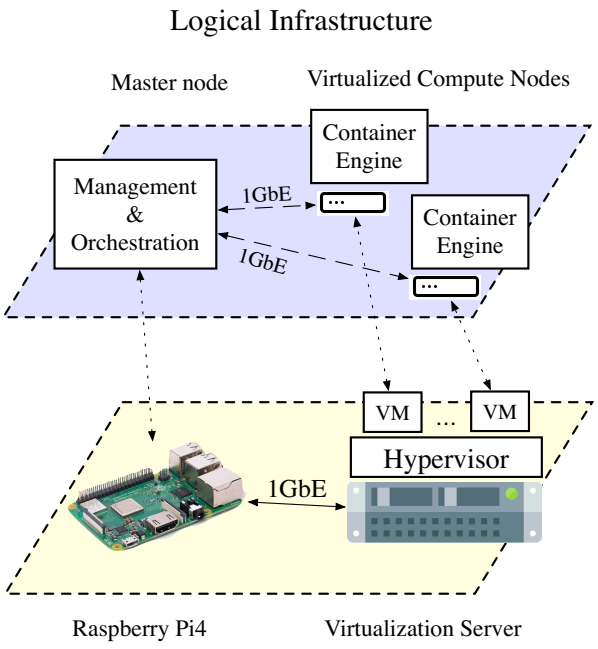


Fig. 3: Illustration of the experimental testbed

solutions, namely OpenStack and Kubernetes, have been designed in accordance with the Cloud computing principles [9]. They provide, respectively, an Infrastructure-as-a-Service (IaaS) and a Platform-as-a-Service (PaaS) over a shared virtualized infrastructure residing in large centralized data-centers and they both consider a logically centralized controller. The controller comprises multiple physical instances of the same software components to address reliability and scalability requirements. Due to the amount of real-time stream-based monitoring information required by controllers to manage the infrastructure [10], a dedicated network for control and management planes is normally provisioned. Moreover, servers are segmented and segregated in clusters to improve resiliency and operational management.

For what concerns Edge computing, multiple solutions have appeared in the market, mainly consisting in an adaptation and extension of existing Cloud solutions to the near-Edge. Such solutions focus on bridging Edge resources under the control of existing Clouds, such as KubeEdge, AWS Outpost or OpenHorizon, de facto extending the Cloud to the Edge. Others instead have put the focus on vertical markets: EdgeX Foundry focuses on IoT applications and their requirements, while Fledge specializes in Industry 4.0 and automation. Still others focus on Edge-only deployments, such as K3s, allowing the management of Edge-only infrastructures without the need of being connected to a Cloud. A brief summary of existing Edge solutions is presented in Tab.I.

Conversely, far-Edge is still a subject of research and no production-ready solutions are available yet. Bearing in mind that far-Edge infrastructure is by nature decentralized, different approaches are being proposed to address such aspect. For instance, [11] analyzes the impact of the far-Edge on vertical markets like Industry automation. Others, like [12], [13] propose to replace the centralized components in nowadays Edge solutions with distributed counterparts. Besides, [14], [15], [16] focus on bridging cloud to support resource constrained devices via ad-hoc allocation policies and reduced resource utilization, with particular attention on how to employ those devices in a mixed scenario of near- and far-Edge.

To better compare Edge and Cloud architectures and solutions, we select Kubernetes as a reference open-source project, which presents tailored versions for the Cloud (i.e., Kubernetes) and the (near-)Edge (i.e., KubeEdge and K3s). It is worth noticing that we selected these projects because: (*i*) there are existing works on extending these solutions for the far-Edge, (*ii*) the selected solutions provide a Cloud-like model to application developers, (*iii*) the selected solutions present a good level of maturity required for the experimental evaluation performed in Sec. IV, and *(iv)* the selected solution are widely used in industry and academia to build Edge computing testbeds.

Fig. 2 illustrates the architectures for Kubernetes, K3s, and KubeEdge. All these architectures are composed of two different sets of components: Control Plane and Compute Plane. The components of the former provide the management and orchestration functionalities across the infrastructure. While the components of the latter provide the deployment and monitoring

functionalities for the nodes composing the infrastructure. A brief overview of each solution is provided in the following.

1) Kubernetes: it is a portable, extensible, open-source platform for managing containerized workloads. It provides a PaaS offering: *(i)* service discovery and load-balancing, *(ii)* storage orchestration, *(iii)* automated rollouts and rollbacks, *(iv)* autonomous placement, *(v)* self-healing and *(vi)* secrets and configuration management. Kubernetes deployments are arranged in clusters. A cluster is a set of nodes that are managed by the same control plane. Kubernetes control plane components are designed around the *kube-api-server* and *etcd*. The *kube-api-server* enables the communication among all other components in the cluster, both control plane and compute plane. While *etcd* provides the storage functionalities for state and infrastructure information across the cluster. These two components are key in Kubernetes as all the interaction goes through them. Kubernetes compute plane is composed by the *kubelet* and the *kube-proxy*. The *kubelet* communicates with the *kube-api-server* for monitoring and container allocation. It manages containers interacting with the container engine. The *kube-proxy* is instructed by the *kube-api-server* about the network configuration needed by the containers.

2) K3s: it is a lightweight version of Kubernetes that keeps the same centralized architecture. k3s comprises two main components: the *k3s-server* and the *k3s-agent*. The *k3s-server* embeds all the Kubernetes Control Plane components, replaces *etcd* with a more lightweight in-process *SQLite* database, and adds networking, monitoring, and load-balacing components. The *k3s-agent* packages in a single binary all the Kubernetes Compute Plane components. It is worth noticing that *etcd* is the most resource demanding component of Kubernetes.

3) KubeEdge: it is an *extension to Kubernetes for Edge hosts*. The goal of KubeEdge is to enable the deployment of applications designed for Kubernetes at the Edge. KubeEdge components are organized in *Cloud components* and *Edge components*. KubeEdge Cloud components are the Control Plane components, they communicate with an external Kubernetes cluster northbound and with KubeEdge nodes southbound. KubeEdge control plane is expected to be deployed in a centralized location. Edge components map to the Compute Plane, they provide functionalities for container deployment as well as an abstraction for well-known Edge devices such as RaspberryPi.

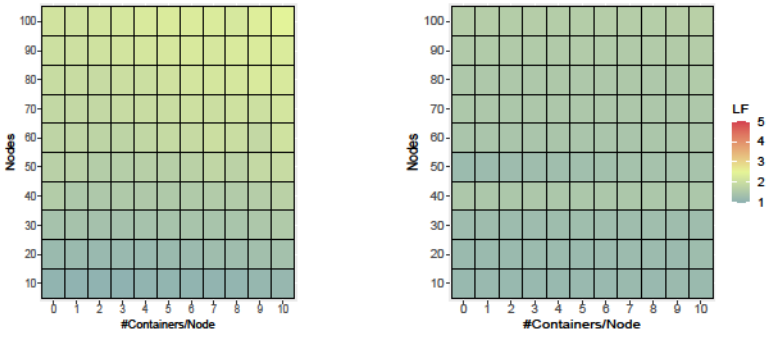
IV. Experimental Evaluation

This section describes the experimental evaluation of the selected Edge solutions: Kubernetes, KubeEdge, and K3s. The first objective of the experimental analysis is to verify baseline deployments based on each selected solution. The second objective is an investigation on their footprint and scalability properties when employed in a far-Edge scenario composed by relatively constrained resources.

A. Setup and methodology

To evaluate the selected solutions, we have built an experimental testbed in our lab as illustrated in Fig. 3. The testbed is composed of a Raspberry Pi4 model B equipped with ARM64 Cortex-A72, 8GB of RAM, 128GB of storage, and 1GbE connectivity that acts as a controller. A rack-mounted server equipped with a Dual Intel Xeon-6130 16 Cores with HyperThreading, 512 GB of RAM, 1TB of SSD storage, is then used to emulate and virtualize 100 compute. Each compute node is virtualized as a VM provisioned with 1vCPU, 2GB of RAM, and 16GB of storage, and Ubuntu 20.04 LTS as the operating system. The selected configuration for the virtualized node is representative of the computing module present inside robots and drones, where typically far-Edge applications will be instantiated. Leveraging on network virtualization technologies our testbed can also introduce arbitrary latency, jitter and packet-loss between each compute node and the controller.

The objective is to measure *(i)* the time required to instantiate a given service and *(ii)* the resources utilized by the controller node to keep the service up and running. This will provide us an indication of how many compute nodes and services can be simultaneously managed in the far-Edge. To that end, we used a reference lightweight service composed of an Alpine-based container running an NGINX web server, which is provisioned before-hand on the compute nodes so as to avoid



(a) Kubernetes (b) K3s (c) KubeEdge

Fig. 4: Average RAM Load factor - $LF(x,y) = \frac{RAM_{x,y}}{RAM_{0,0}}$

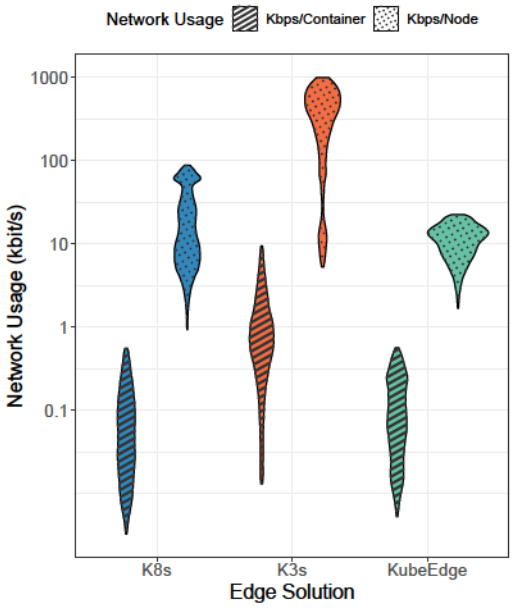


Fig. 5: Network Usage

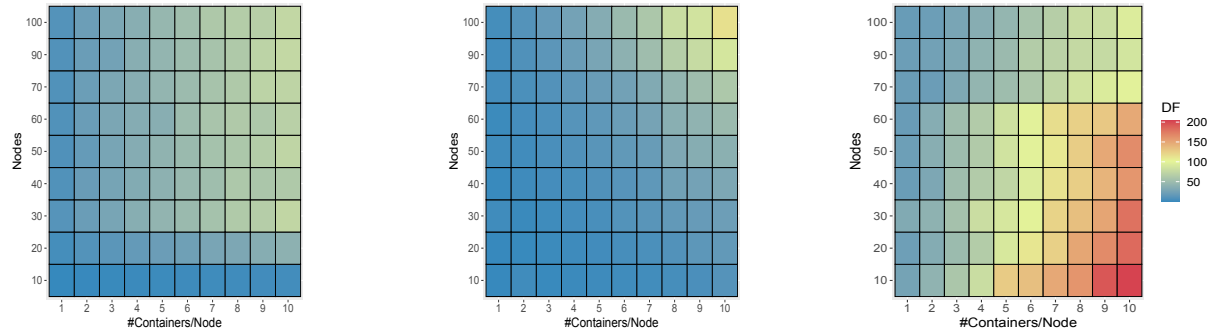
copying the container images over the network during the deployment. To test the scalability, we consider an increasing number of compute nodes (from 10 to 100 with a step of 10 nodes) while on each node an increasing number of services being deployed (from 1 to 10 with a step of 1 service). Each test is then performed 10 times. Given that the communication between compute nodes and controller is based on TCP, introducing latency, jitter, packet-loss or bandwidth limitation, between the operative limits of TCP will not affect the communication between the nodes.
However in a real-world scenario, where the images are not prefetched, having latency, jitter, packet-loss and bandwidth limitation has an huge impact in the application instantiation time because the instantiation time will be dominated by the image fetching time.

B. Results

Fig. 4 illustrates and defines (in the caption) the RAM load factor measured in the controller node at the variation of compute nodes and containers/node. This metric provides the amount of memory variation between a baseline (i.e., an empty cluster), while increasing the number of machines controlled, and the number of applications running in the cluster in order to facilitate the comparison between the different solutions. While each solution presents different absolute values of RAM usage, we remark that in every case the 5^{th}-95^{th} percentiles never exceed $\pm 2\%$ of their corresponding median.

In Fig. 4 we can see that for a cluster of 10 nodes with 0 containers, Kubernetes uses 27% more RAM compared with K3s, and 9% more compared with KubeEdge, while KubeEdge uses 17% more RAM than K3s. If we increase the number of nodes in the system to 100 with 0 containers, we can notice that Kubernetes uses 91% more RAM compared with K3s, and 143% more than KubeEdge. In such a scenario Kubernetes, K3s, and KubeEdge show, respectively, an increase of RAM usage of 218%, 110% and 42% compared with an empty cluster(0 nodes with 0 containers).

We observe that the number of nodes composing the cluster is the major factor impacting the overall



(a) Kubernetes (b) K3s (c) KubeEdge

Fig. 6: Average Deployment factor - $DF(x,y) = \frac{T_{Deploy[x,y]}}{T_{Deploy[K8s][1,10]}}$

memory utilization. This is due to need of the controller to store large amount of information about the state and available resources for each node in the cluster. Finally, when we increase the number of containers per node (e.g., 100 nodes with 10 containers per node), Fig. 4 shows marginal increase of 6% for Kubernetes, while K3s and KubeEdge RAM usage increases for 23% and 5%, respectively. On the contrary, we observe that increasing the number of applications running on the cluster has marginal impact on the memory utilization. This is because the runtime information (e.g., container state, networking information) are stored in the compute nodes and not in the controller, which in turn keeps track only of the containers location and retrieves any additional information only when needed.

Regarding network utilization, Fig. 5 shows that the maximum network utilization per node is approximately two orders of magnitude higher then the network utilization per container. K3s shows a network utilization per node of 30 times higher than Kubernetes and KubeEdge. Moreover, it shows a higher network utilization per container of approximately 10 times compared to the other two solutions. This is partially due to the Metrics Server that K3S uses as one of its packaged components. The Metrics Server reads 10 metrics for each container and node running in the cluster. Those metrics are used in horizontal and vertical autoscaling provided by K3s.

Fig. 6 illustrates and defines (in the caption) the Deployment factor measuring the variation of the service instantiation time at the variation of compute nodes and containers/node. Each tile in the figure represents the Deployment factor. This metric allows comparing the different solutions by normalizing the absolute value of the instantiation time to the fastest one, i.e. Kubernetes. In Fig. 6, we observe that Kubernetes and K3s show similar and intuitive growth of the instantiation time when the overall number of containers in the cluster is increased. On the other side, KubeEdge presents a very different and counter intuitive pattern. For deploying a service composed of 10 containers per node in a cluster with 10 nodes, KubeEdge needs approximately 40 times more than Kubernetes and 18 times more than K3s to instantiate the service. The overall instantiation required in KubeEdge ranges from 26.1 to 40.4 times longer than Kubernetes and from 18.1 to 21.1 times longer than K3s. In absolute terms, this translates into tens of minutes instantiation time with KubeEdge compared to tens of seconds in Kubernetes and few minutes in K3s. This is because, with a limited number of nodes in the cluster, KubeEdge tends to allocate all the containers in a subset of the available nodes, resulting in the nodes being overloaded and a longer overall service instantiation time.

Summarizing, from the presented results we observed that:

1) the way solution component are deployed (containerized or simple binaries) influences memory utilization;
2) memory utilization is dominated by the number of nodes composing the infrastructure;
3) monitoring of the infrastructure requires

Edge Solution	Cloud Managed	Edge only	Focus
KubeEdge	Yes	Possible	Containers
K3s	No	Yes	Containers
AWS Outpost	Yes	No	VMs Containers
OpenHorizon	Yes	No	Containers
EdgeX Foundry	Yes	No	IoT
Fledge	Yes	No	I4.0

TABLE I: Summary of exiting Edge Computing solutions

approximately two orders of magnitude more bandwidth compared to container monitoring;
4) scheduling decisions may overload some nodes increasing the overall service instantiation time.

V. Discussion

This work discusses the issues of today's centralized management solutions when applied to the far-Edge; an environment heavily characterized by volatility, mobility and decentralization of resources. During our analysis we have identified several aspects to consider in the design of the next generation Edge systems:

a) Infrastructure monitoring: it plays a critical role in the management systems where the life-cycle management of the applications highly depends on the real-time information about the state of the infrastructure. To support this information, current management systems (see Sec. III) adopt a stream-based monitoring approach, which poses stringent requirements on networking, like always-on connectivity, minimum required bandwidth and low-latency. However, based on our analysis of the far-Edge infrastructure in Sec. II, this scenario is characterised by unreliable networking without any guarantees. Hence, a stream-based approach can led to saturation of the already scarce networking resources, resulting in a harmful behavior. In addition, the use of outdated information in real time decisions, e.g. scheduling, may yield to sub-optimal performance. A different approach is then needed to overcome this issue like new monitoring paradigms designed to work over low-bandwidth and unreliable networks. Examples of such paradigms are event-based and threshold-based monitoring, which relax requirements over the network.

b) Information model: is the representation of concepts, relationships, constraints, rules and operations used to describe a certain domain. In our case it refers to how each solution describes the underlying infrastructure, applications and their relationships. Based on our analysis in Sec. III current solutions are designed for the Cloud environment, composed of static, large scale data-centers (see Sec. II). Moreover, Cloud applications are usually packaged either as Virtual Machines or containers. While these virtualization technologies provide isolation for applications sharing the same infrastructure, they introduce a limitation on the features they can access on the host machine. Far-Edge Applications in use-cases like Edge robotics, UAVs and vehicles may require direct access to sensors and actuators that for security reasons should be given via privileged containers/VMs. This is an example of new characteristics introduced by the far-Edge concept. These characteristics are very different from the ones that can be found in a static data-center, as a consequence they are not considered in current management solutions. Therefore, extensions to the current information model should be considered in order to: *(i)* keep track of the different capabilities, and mobility characteristics of underlying infrastructure and *(ii)* allow broader application packaging technologies (e.g., robotic applications, micro-controller applications).

The next issue identified deals with the consistency of the system and its degree of centralization. Before discussing them, it is worth introducing the CAP theorem and its implications in both the Cloud and Edge domains. The CAP (Consistency, Availability and Partition) theorem was introduced in [17], and it states that any distributed system that has to deal with data can only provide two of the three following characteristics:

- *Consistency*: everyone who reads information from the system gets either the most recent information or an error.
- *Availability*: everyone who reads information gets a (non-error) response without the guarantee that this is the most recent one.

- *Partition tolerance*: the system continues its operations despite messages being dropped by the network connecting the system elements.

Current Cloud solutions cope with the problem described by this theorem by leveraging on high-redundant networking and replication of data and services (see Fig. 1). This approach allows to reduce at minimum the risk of partitioning and therefore such systems can operate by choosing Consistency and Availability as their guarantees, with the Partitioning provided by the underlying physical infrastructure. However, in the far-Edge environment it is unfeasible to reach such level of physical redundancy, resulting in a high risk of system partitioning that must be considered. To get a trade-off between the three guarantees in the far-Edge scenario, it is needed to explicitly handle partitioning [18]. This need goes against the centralized control and systems state considered in Cloud environments.

c) Centralized control plane: as illustrated in Fig. 2 each of the analyzed solutions is based on a centralized control plane. Such design is common in Cloud solutions (see Sec. III) because it facilitates development. To achieve high-availability and consistency, replication techniques, inherited from state of the art web approaches can be used. However, as state above, to achieve such properties in the far-Edge it is necessary to explicit handle partitions. This means that control functions like scheduling and monitoring need to be aware of the possible partitioning of the system, and take decisions involving the trade-off between consistency and availability.

In the light of this, we believe that distribution of control functions and intelligence among nodes in the system should be considered, because: *(i)* each node can intelligently react to events even if it is partitioned and *(ii)* the whole system does not stop its evolution in case of partition.

d) Centralized system state: current solutions are based on a centralized storage for the whole system state that acts as source of truth (see Sec. III). This approach is common in Cloud and near-Edge since they can maintain easily a centralize repository of information. However, as shown by experimental results from Sec. IV this requires a good amount of system memory devoted to the storage of state information, yielding to the need of powerful central servers to store and provide this information across the whole system. However, in the far-Edge environment *(i)* redundancy may not be applicable, *(ii)* such powerful machines may not be present and *(iii)* partitions are likely to happen. The state information is crucial for the system as all the intelligence and decisions are based on them.

Therefore, distributing the state information across nodes participating in the system should be considered since *(i)* it reduces the burden of having one single information storage requiring large memory quantities by distributing the memory needs across the nodes composing the system and *(ii)* enables partitioning while providing availability.

It is clear that this approach needs to be handled by design as it may require a change on the consistency model of the system, that must be taken into account when designing the management algorithms.

VI. Conclusions

Recent years saw the advent of different Edge solutions, each one focusing on a different aspect of the Edge. This work presents an analysis of the differences between Cloud and Edge infrastructure and the most prominent current solutions for controlling edge resources, an evaluation on the scalability properties is also provided. Results show that, *(i)* the amount of information stored in the controller is dominated by infrastructure information, *(ii)* infrastructure monitoring relies on a constant stream of information that may be incompatible with far-Edge connectivity, *(iii)* scheduling decisions can lead to over-provisioned nodes, leaving part of the infrastructure unused, and, *(iv)* existing solutions support only containerized applications workloads, whereas with far-Edge an heterogeneous approach is needed. As next steps, we plan to investigate the proposed areas, in particular the modeling of a decentralized, heterogeneous, mobile, and volatile infrastructure and management solutions that do not require centralized controllers.

VII. Acknowledgement

This work has been partially funded by the H2020 collaborative Europe/Taiwan research

project 5G-DIVE (grant no. 589881) and by the H2020 European collaborative research project DAEMON (grant no. 101017109).

REFERENCES

[1] S. Liu, L. Liu, J. Tang, B. Yu, Y. Wang, and W. Shi, "Edge computing for autonomous driving: Opportunities and challenges," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1697–1716, Aug 2019.

[2] ETSI, "Multi-access Edge Computing (MEC); Terminology," European Telecommunications Standards Institute (ETSI), Group Specification (GS) 001 v2.1.1, 1 2019.

[3] ——, "Network Functions Virtualisation (NFV); Use Cases," European Telecommunications Standards Institute (ETSI), Group Report (GR) NFV 001 v1.2.1, 5 2017.

[4] IEEE, "Standards for Adoption of OpenFog Reference Architecture for Fog Computing," Institute of Electrical and Electronics Engineers (IEEE), FOG - Fog Computing and Networking Architecture Framework 1934, 7 2018.

[5] Networld2020, "Smart Networks in the context of NGI," Networld2020, Tech. Rep., 4 2018.

[6] I. Goiri, K. Le, J. Guitart, J. Torres, and R. Bianchini, "Intelligent placement of datacenters for internet services," in *2011 31st International Conference on Distributed Computing Systems*, June 2011, pp. 131–142.

[7] Ramneek, S. Cha, S. H. Jeon, Y. J. Jeong, J. M. Kim, S. Jung, and S. Pack, "Boosting edge computing performance through heterogeneous manycore systems," in *2018 International Conference on Information and Communication Technology Convergence (ICTC)*, Oct 2018, pp. 922–924.

[8] C. Hong and B. Varghese, "Resource management in fog/edge computing: A survey," *CoRR*, vol. abs/1810.00305, 2018. [Online]. Available: http://arxiv.org/abs/1810.00305

[9] NIST, "The NIST Definition of Cloud Computing," National Institutee of Standards and Technology, Tech. Rep. NIST Special Publication 800-145, 10 2011.

[10] C. Wang, K. Schwan, V. Talwar, G. Eisenhauer, L. Hu, and M. Wolf, "A flexible architecture integrating monitoring and analytics for managing large-scale data centers," in *Proceedings of the 8th ACM International Conference on Autonomic Computing*, ser. ICAC '11. New York, NY, USA: Association for Computing Machinery, 2011, p. 141–150. [Online]. Available: https://doi.org/10.1145/1998582.1998605

[11] M. Ciavotta, M. Alge, S. Menato, D. Rovere, and P. Pedrazzoli, "A microservice-based middleware for the digital factory," *Procedia Manufacturing*, vol. 11, pp. 931 – 938, 2017, 27th International Conference on Flexible Automation and Intelligent Manufacturing, FAIM2017, 27-30 June 2017, Modena, Italy. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2351978917304055

[12] A. Lebre, J. Pastor, A. Simonet, and F. Desprez, "Revising openstack to operate fog/edge computing infrastructures," in *2017 IEEE International Conference on Cloud Engineering (IC2E)*, April 2017, pp. 138–148.

[13] Y. Xiong, Y. Sun, L. Xing, and Y. Huang, "Extend cloud to edge with kubeedge," in *2018 IEEE/ACM Symposium on Edge Computing (SEC)*, Oct 2018, pp. 373–377.

[14] P. Kayal, "Kubernetes in fog computing: Feasibility demonstration, limitations and improvement scope : Invited paper," in *2020 IEEE 6th World Forum on Internet of Things (WF-IoT)*, 2020, pp. 1–6.

[15] T. Goethals, F. DeTurck, and B. Volckaert, "Extending kubernetes clusters to low-resource edge devices using virtual kubelets," *IEEE Transactions on Cloud Computing*, pp. 1–1, 2020.

[16] A. Munir, P. Kansakar, and S. U. Khan, "Ifciot: Integrated fog cloud iot: A novel architectural paradigm for the future internet of things." *IEEE Consumer Electronics Magazine*, vol. 6, no. 3, pp. 74–82, 2017.

[17] E. A. Brewer, "Lessons from giant-scale services," *IEEE Internet Computing*, vol. 5, no. 4, pp. 46–55, 2001.

[18] E. Brewer, "Cap twelve years later: How the "rules" have changed," *Computer*, vol. 45, no. 2, pp. 23–29, 2012.

Gabriele Baldoni received his Bachelor's and Master's degree in Computer Engineering and Telecommunication Engineering at the University of Catania, Italy, in 2015 and 2017 respectively. From 2017 works as research engineer at ADLINK Technology. Starting from 2020 he is a PhD Student at University Carlos III of Madrid (UC3M) his main focus is end-to-end orchestration in Edge and Fog environment. Contact him at gabriele.baldoni@adlinktech.com

Luca Cominardi received his Bachelor's and Master's degrees in Computer Science at the University of Brescia, Italy, in 2010 and 2013, respectively. He received his Master's degree and Ph.D. in Telematics Engineering from the University Carlos III of Madrid (UC3M), Spain, in 2014 and 2019, respectively. From 2019 works as Senior Technologist at ADLINK Technology working on Fog Computing and distributed systems. Contact him at luca.cominardi@adlinktech.com

Milan Groshev received the B.S. degree in telecommunication engineering from the The Saints Cyril and Methodius University of Skopje, Macedonia in 2008 and the M.S. degree in telecommunication engineering from the Politecnico di Torino, Turin, Italy in 2016. He is currently pursuing the Ph.D. degree in telematics engineering at University Carlos III Madrid (UC3M), Spain. Contact him at mgroshev@pa.uc3m.es

Antonio De La Oliva received his telecommunications engineering degree in 2004 and his Ph.D. in 2008 from the Universidad Carlos III Madrid (UC3M), Spain, where he has been an associate professor since then. He has served as Vice-Chair of IEEE 802.21b and Technical Editor of IEEE 802.21d. He has also served as a Guest Editor of IEEE Communications Magazine. Contact him at aoliva@it.uc3m.es

Angelo Corsaro is Chief Technology Officer (CTO) at ADLINK Technology Inc. where he looks after corporate technology strategy and innovation, leads the Advanced Technology Office and the Software and Technology Business Unit. He is a world top expert in edge/fog computing and a well-known researcher in the area of high-performance and large-scale distributed systems. Contact him at angelo.corsaro@adlinktech.com