On the Efficiency of Service and Data Handoff Protocols in Edge Computing Systems

Domenico Scotece†, Claudio Fiandrino⋆, Luca Foschini†
†Department of Computer Science and Engineering, University of Bologna, Bologna, Italy
⋆IMDEA Networks Institute, Madrid, Spain
Email:†{domenico.scotece, luca.foschini}@unibo.it
⋆claudio.fiandrino@imdea.org

***Abstract*—The Multi-access Edge Computing (MEC) enables a new layer of edge middleboxes, acting as local proxies with virtualized resources deployed at edge localities. To support scalable, low-latency, and locally managed service provisioning, MEC relies on computation offloading, the process that outsources computing tasks from resourced constrained mobile devices and moves it to edge data centers. In this paper, we tackle a specific sub-problem within the umbrella of computation offloading. We argue that it is convenient to migrate a service because of the lack of computing resources in the anchor edge data center even if a device, such as industrial IoT devices, is not moving. In this paper, we extensively evaluate the efficiency of data and service handoff protocols. Specifically, we thoroughly assess protocols, that we designed in our past work, in a well-known edge computing emulator, i.e., openLEON. These protocols migrate data and service either in a reactive fashion, i.e., upon realizing of resource exhaustion, or proactively, i.e., beforehand to swiftly minimize the downtime. We experimentally verify their performance for a typical MEC use case, i.e., video. Our results show that by being proactive, the service interruption downtime reduces by a factor of 4 times.**

***Index Terms*—Multi-access Edge computing, Computation Offloading, Service Migration, IIoT, Docker Container**

I. Introduction

In the recent years, edge computing has gained momentum and attracted the interest of both industry and academia [1], [2]. Driven by the widespread diffusion of mobile and wearable devices and the tremendous growth of the Internet of Things (IoT), a number of new applications has emerged all calling for stringent requirements such as intensive computation and tight latency budgets. Examples of such applications are virtual and augmented reality (VR/AR), navigation, gaming among the others [3]. Edge computing is key to meet such stringent constraints by pooling computing resources closer to the end user and not in the cloud. Such concept resulted into various paradigms, including fog computing, mist computing [4], and Multi-Access Edge Computing (MEC) [5]. The latter was standardized by the European Telecommunications Standards Institute (ETSI) [6] and is specifically tailored as an enabler for the 5th Generation (5G) mobile networks and beyond.

The growth of edge computing enables the distribution of computing from centralized datacenters to small distributed datacenters at the edge of the network. Computation offloading [7] is the process that moves away from the resource-constrained mobile devices part of the computing tasks onto the edge, e.g., a nano data center deployed in close proximity of cellular base stations [8]. This is key to cut the latency and augment performances of the mobile devices by prolonging battery life time [9] and enabling the execution of applications that would be impossible to run on the mobile devices alone [10]. In the presence of mobile users, offloading is more complex because the process of detaching from the current service anchor (base station and server) and attach to the new anchor increases latency. This negatively impacts user Quality of Experience. To overcome such limitation, migrating the service anchor in a way that follows the user movement has been proven effective [11], [12].

The Industrial IoT (IIoT) is an evolution of the IoT concept that aims at handling real-time transmission of big data, with reduced cost, improved latency, and robust connectivity. In addition, in IIoT real-time decisions should result in higher efficiency, safety, and should guarantee stability even in large scale deployments. IIoT is regarded a promising enabler for Industry 4.0 and has been widely studied and employed in various scenarios [13]. Essentially, IIoT devices are deployed as an integrated infrastructure that collects information from heterogeneous sensors, delivers it to edge datacenters, and updates all the related parameters of the closed-loop system [14]. For an IIoT platform to be effective, the closed-loop system has to be managed in a timely and efficient way in order to satisfy the strict requirements of IIoT services. For that reason, there is a need for efficient migration strategies when computational resources at the edge fail during the execution of delicate tasks. Specifically, it is crucial to proactively identify when an edge node is running out of resources for this kind of application (e.g., with computing and networking benchmarking indicators [15]). Moreover, granting service continuity is considered a vital issue for IIoT systems, and migrating a service to another edge datacenter can allow the proper execution of the service itself although with an acceptable delay. Therefore, we claim that a study, with a real emulator approach, is crucial to confirm and assess the performances of service migration support.

Accordingly, in this paper, we consider two service and data handoff protocols proposed in our previous work [16] that enable migration when computational resources available at the edge become scarce. The protocols support two different migration approaches. The *reactive* one performs migration only when the handoff occurs (i.e., after resource exhaustion). By contrary, the *proactive* protocol provides resource pre-allocation

mechanisms that start service and data migration prior to the occurrence of the handoff. Specifically, we take the research on data and service handoff one step further by evaluating the performance of the aforementioned protocols with an edge emulator. The protocols are designed to support MEC/fog-enabled handoff management with the migration of either VM-based or container-based virtualized resources. Specifically, in this work, we exploit container-based technologies, such as Docker that features well-known tangible benefits against VMs in several deployment and application scenarios. We then consider a video streaming application, representative of typical MEC use case scenarios, by implementing reactive and proactive protocols in the openLEON emulator [17]. openLEON allowed to benchmark the performance of edge solutions end-to-end, from the edge data center to the end mobile user or device.

In a nutshell, the flow of the paper is as follows. First, we provide our motivation for this study and related works (Section II). Then, we present our service/data handoff protocols that are applicable to tasks that are sensitive to service continuity like process optimization, quality inspection, and preventive diagnostics in IIoT scenarios (Section III). Finally, we benchmark the algorithms with openLEON (Section IV) because the lack of experimental results from prototypes is often indicated as a primary shortcoming of the current MEC/fog literature [18].

II. Background and Motivation

A. Background

This section provides background information for the involved technologies and paradigms and briefly introduces the research directions of the literature in the areas of edge computation offloading and service migration to the edge.

1) Offloading Computing at the Edge: Offloading computation to the edge has attracted extensive research in the past years because of the inherent advantages it can offer (e.g., lower latency and jitter). For the IoT ecosystem, offloading computation at the edge also brings considerable advantages in terms of quality of service [19]. On the one hand, computation offloading at the edge has been applied to Industrial IoT [20] in order to reduce the delay in typical industrial closed control loop applications. On the other hand, computation offloading oriented optimization is also proposed, such as the energy-efficient computation offloading and resource allocation [21], for prolonging the battery life and enhancing the computational capacity of mobile nodes. Finally, another relevant scenario for computation offloading is edge-cloud collaboration. This scenario has been studied in [22], [23]. In a nutshell, edge computing and cloud computing will collaborate in several environments including smart cities, smart homes, industrial IoT, connected autonomous vehicles, and so on, to take full advantage of on-premises edge and cloud computing capabilities. Our proposal, even if is not directly related to computation offloading, complements computation offloading at the edge by solving a sub-problem which is *moving the computation* from an edge data center to another one if this helps to guarantee an acceptable end-to-end delay.

2) Service Migration at the Edge: More in general, existing literature about handoff at the edge has primarily concentrated on live migration aspects: some proposal focused on VMs live migration [24]; a very few solutions were presented about container migration, such as [25] that proposed to use containers in its live service migration framework by showing the associated advantages, or IBM Voyager [26] that proposed a live container migration service, which is vendor-agnostic, with consistency guarantees, and designed according to the Open Container Initiative (OCI) principles. Highly relevant to this research area is the work by Satyanarayanan et al. [27] that proposes a mechanism for edge-enabled handoff management based on VM service synthesis and migration to the newly visited edge nodes. However, the use of VMs adds complexity, i.e., it introduces a non-negligible latency and overhead to handle the VM size. Furthermore, L. Ma et al. [28] reduce filesystem synchronization time that is critical during the handoff by leveraging the layered nature of Docker containers. The results show a quantitative comparison between Docker container migration and monolithic services VM migration.

However, most of the past works mentioned above, are typically based on VMs and/or on a reactive strategy. This is not suitable for a wide range of emerging applications that require a high amount of data processing and low service downtime. Sometimes, resources available at the edge could be not enough to allow sophisticated migration mechanisms (just think of IoT networks with Fog nodes as smart gateways), which means the need of lightweight migration system and efficient in terms of time and resource consumption. In conclusion, most of the related works lack in-the-field experiments and, to the best of our knowledge, provide only comparisons of different migration mechanisms and/or different virtualization technologies. The latter point motivates our work. In fact, we provide a rich set of in-the-field experiments, which allow us to calculate the real goodness of our solution in real networks. Finally, in this paper, we evaluate two migration mechanisms that supports both reactive and proactive container migration. Moreover, it based on the resource available at the edge and leverage two different handoff trigger mechanisms. Finally, we believe that this paper provides a relevant contribution to the community because, to the best of our knowledge, it is one of the first system-oriented prototypes that works on 5G-enabled edge emulation testbed.

B. Motivation

We illustrate the need for service/data migration in IIoT environments with the example shown in Fig. 1. Devices need to run heavy computations and decide to execute services at the edge layer by leveraging virtualized resources that it provide. Note that the edge datacenters are all at the same level behind the same 5G base station. The proximity between mobile devices and edge datacenters allows for low latency and better reliability, especially, in the IIoT environment. After some time, during service provisioning, edge datacenter1

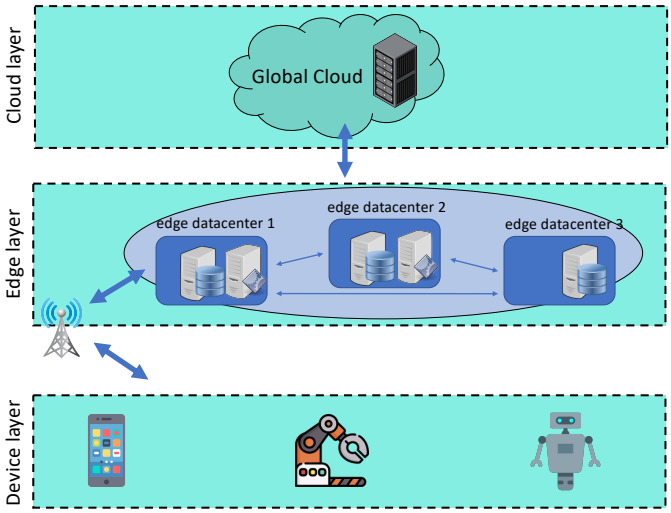


Figure 1. An example of IIoT edge-enabled architecture

consumes all resources (i.e., CPU and memory) and is no longer able to provide services. In order to grant service continuity, the infrastructure should be able to transparently migrate service/data components from the edge datacenter1 to the new edge datacenter2, which is considered more unloaded, by possibly also guaranteeing continuous service provisioning without interruptions. Note that edge nodes should also benefit from some form of lightweight coordination among them. This does not require coordination from the cloud and it is based either on their hierarchy (to support scalability and wide-scale deployment) or on peer-to-peer interactions.

For instance, Unmanned Aerial Vehicles (UAVs) are commonly used in the industry sector for goods evaluations purposes, such as object recognition which requires stable wireless connectivity and local computing power at the edge, with the use of wireless and 5G technologies. In our architecture example, a UAV connects to industrial control services that provide object recognition task over the 5G network. To ensure low latency, the UAV is served by a closer base station and the computation by a single datacenter. The problem that we want to tackle is the re-location of computation to guarantee system high availability. Indeed, a problem may occur if a UAV is connected to a base station and attached to the anchor edge datacenter1: for instance, for the exhaustion of CPU, we move one "run-execution" to the edge datacenter2 in order to guarantee service availability under certain quality of services. In particular, our vision is to proactively identify edge datacenters congestion in a way that we have the service running on an edge datacenter which ensures the quality of service.

III. Efficient Data Handoff at the Edge

This section introduces our algorithms for service and data migration in edge computing systems. First, we present the simple reactive protocol followed by our proactive optimization [16]. In a nutshell, when the handoff is predicted to occur, the proactive approach moves beforehand service and data towards the destination edge datacenter. Next, only when the handoff actually occurs, the proactive protocol starts the service and moves the remaining data at the destination edge datacenter.

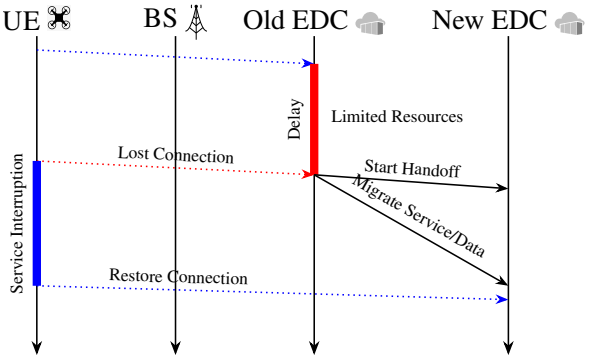


Figure 2. The baseline reactive handoff protocol

A. Reactive Handoff Protocol

Fig. 2 shows the workflow of the reactive procedure. The handoff starts when the performances at the old edge datacenter are deteriorated according to a specific model. The worsening performances at the old edge datacenter mean that the node is suffering delays for its request. Let us note that handoff decision logic, to determine when and where to perform handoffs (i.e., handoff triggering), is depending on several aspects including resource availability at edges. Most handoff triggering algorithms in the literature are based on user mobility patterns. However, in this work, we distinguish two kinds of handoff triggering: a (short-term) handoff triggering that predict the handoff in the immediate; a (long-term) handoff triggering which can effectively enable longer handoff management operations such as proactive pre-loading of some static heavy content (e.g., service layer code) at the target edge. Both the handoff triggers monitor resources available at the edge and start the handoff procedure when resources are under a specific threshold. Handoff trigger algorithms are out of the scope of this paper. Therefore, we are working on mathematical models to manage efficiently the handoff process.

After handoff triggers, the old edge datacenter starts the migration process involving both service and data software layers towards the target edge node (black continue lines) while the connection with the node is interrupted (red dashed line). In this work, we leverage virtualization technologies, in particular, Docker Container which allows us to move software resources and restore them easily with the help of an orchestrator, for instance, Kubernetes. Let us observe that Fig. 2 describes our baseline reactive handoff protocol, which introduces a service interruption that involves all steps (blue block in the bottom left); we have implemented it in order to compare it against our optimized proactive handoff protocol.

B. Proactive Handoff Protocol

The core idea of our optimized handoff algorithm is to proactively move service and associated data parts from an edge datacenter to another one in order to keep an acceptable service downtime as a consequence of the service migration.

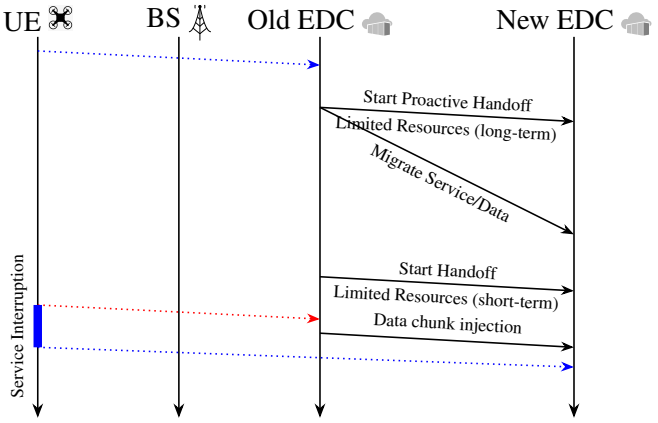


Figure 3. The proactive handoff protocol

For this to happen, it is important to: i) identify potential exhaustion of resource at an edge datacenter early (i.e., long-term handoff trigger strategies), and ii) migrate service and data part proactively and, if necessary, sync small data chunks once the handoff happens.

Fig. 3 shows the workflow of our proactive handoff protocol. In the beginning, the user node executes the service at the old edge datacenter (blue dashed line). The handoff starts when a long-term trigger fires the handoff signal and the old edge datacenter starts to proactively migrate the service and the data software layers towards the new edge datacenter. These steps can be done by exploiting standard orchestrators that help to move virtualized resources among datacenters. Note that the user node is still hocked at the old edge datacenter for the computation execution. Then, when the old edge datacenter goes effectively down, so after the short-term handoff trigger, the handoff procedure can be completed, and the service is hooked at the new edge datacenter. The red dashed line, shown in Fig. 3 indicates the termination of the service at the old edge datacenter while the last blue dashed line indicates the service restart at the new edge datacenter. These lines determine the service interruption time. Finally, a further phase called data chunk injection guarantees data consistency between the two edge datacenters. This might happen when some data is changed during the handoff. In order to achieve minimum service downtime, the proactive handoff protocol leverages both long and short terms handoff trigger mechanisms.

IV. Performance Evaluation

This section provides performance evaluation of the handoff strategies discussed in Section III. We first discuss the experimental setup of openLEON and next we discuss the set of comprehensive results.

A. Experimental Environment

Fig. 4 shows the openLEON setup. To run the data center network emulated with Containernet [29] and the core network (`srsEPC` application from srsLTE version 19.0.6) we use a laptop equipped with an Intel i7-4600U processor at 2.1 GHz, 8 GB RAM and Linux Ubuntu 16.04 LTS. Then, to run the the BS application (`srsENB` application from srsLTE version 19.0.6),

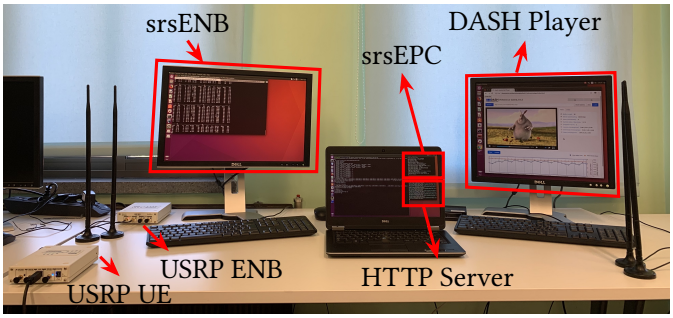


Figure 4. The openLEON platform. In this example setup, the UE consumes a video obtained from a server in the edge. For our experiments, we set the stream direction the other way round.

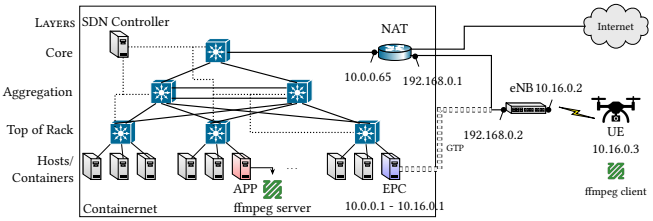


Figure 5. The generic openLEON architecture to test the `ffmpeg` application

we use a desktop computer equipped with an Intel i7-6700 processor running at 3.4 GHz, 16 GB RAM and Linux Ubuntu 16.04 LTS. The physical BS is an Ettus B210 USRP connected with USB 3.0 to the desktop computer. The UE as well uses a Ettus B210 USRP and it is connected to a laptop with an Intel i7-4600U processor up to 2.1 GHz and 8 GB of RAM that executes the UE application (`srsUE` application from srsLTE version 19.0.6). All results obtained in the emulation are an average of about 30 runs and have exhibited a limited variance, i.e., below 5%.

To assess the performance of proposed protocols, we devise an Industry 4.0 application aligned with one of the ETSI MEC use cases, such as video analytics. Specifically, we focus a scenario where a group of UAVs in a factory are connected to the 5G infrastructure to perform decentralized decision-making, for instance, object recognition tasks. To emulate this, we have created a virtualized network with the Containernet simulation tool which creates the network topology and the edge datacenters. Fig. 5 shows the architecture topology that we implement in openLEON for preliminary tests: no migration is enforced here. The figure provides an intuition of the network setup that is required to operate such a edge scenario. For the video streaming task, we use the well known `ffmpeg` software that is a Linux-based standard tool for video/audio streaming.

To assess the performance of the proposed protocols we need to re-architect out setting to allow for migration of the Dockerized `ffmpeg` application and its associated data, including state and recognized objects. The UE is always anchored at the base station (the `eNB` node in the Fig. 6). At the beginning, the UE streams the video to the server located in the edge datacenter1 (EDC 1 in the figure). During the handoff, the service and associated state is moved from

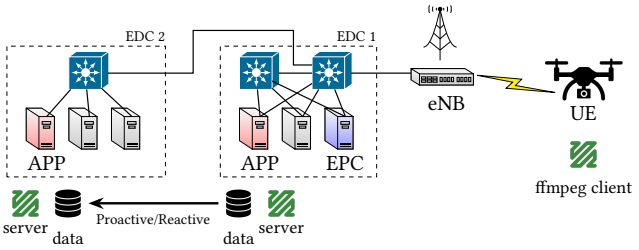


Figure 6. The openLEON architecture used for performance evaluation

the edge datacenter1 to the edge datacenter2 (EDC 2 in the figure) in a way that is transparent to the UE. In this work, for the sake of simplicity and without loss of generality, we assume that the edge datacenter1 starts the handoff process depending on the current load of the edge datacenter, measured as CPU consumption with the linux `sysstat` tool. For fair comparison, we consider the same threshold value for both handoff types.

B. Experimental Results

With the experiments, we analyze the impact of the proposed handoff algorithms on service continuity. Specifically, we measure the service interruption time and the average throughput during the handoff procedure.

In the first experiment, we assess the baseline reactive handoff protocol (see Fig. 2). The handoff process is started when the CPU consumption of the edge datacenter1 is above a certain threshold. In such case, the entire volume of the service layer software and the data state to be migrated is around 20 MB. The total duration of the service interruption during migration from the edge datacenter 1 to the edge datacenter2 that is observed by the mobile UE depends on the interval needed for all migrations and startups. Note that all procedures mentioned above are completely transparent to the UE who keeps connected to the very same base station. We set up a service proxy at the base station to allow service redirection when the handoff is completed. The associated performance results for reactive handoff are reported in Fig. 7(a) which shows the channel transmission between the UE node and the base station. When the handoff occurs, the communication flow between the UE node and the service is interrupted. Then, the communication flow restarts again only when the handoff and all startups at the edge datacenter2 are completed. The total service interruption time in the case of the reactive handoff is around an average of 9 s. Note that the time reported in Fig. 7 is an observed time window for experiment purposes.

We now focus to the more interesting proactive scenario. In the second experiment, we assess the proactive handoff procedure (see Fig. 3). Fig. 7(b) shows the corresponding results. When the long-term trigger predicts the handoff, the Docker images for service and data containers are proactively installed on the edge datacenter2. Therefore, this first phase of the algorithm is performed only when the CPU consumption of the edge datacenter1 is above the first critical threshold which suggests that the node is going to fail. This migration is completely transparent to the device. Then, when the CPU consumption of the edge datacenter 1 is above the second critical threshold, the handoff takes place. To continue the communication flow the UE node has to wait until the service restarts at the edge datacenter2. Note that in this way, in this specific case, we have reduced the need for data exchange during the handoff. Service static data were sent before the handoff. Overall, the proactive scenario has shown to drop down the service interruption time to around 80% compared to the reactive handoff.

Fig. 8 (a) shows the service downtime for both algorithms while Fig. 8(b) shows the throughput observed on the link interconnecting the device and the edge server. The results are averages over 30 runs with corresponding 95% confidence intervals. Both algorithms ensure an acceptable average throughput even during the data handoff. Finally, Fig. 8(c) shows the average Round Trip Time (RTT) observed in the time window for both handoff algorithms. Similarly to the throughput analysis, the average RTT is lower in the case of the proactive handoff. A lower RTT is crucial for latency-sensitive applications, hence using the proactive mechanism provides an advantage. The analysis of the packet traces that we collect unveil that for the reactive case, besides zero-window and keepalive packets present during the downtime to test if the connection is still up, we also observe many more retransmissions-time-outs (RTOs) which make significantly longer the delays for retransmitting lost packets.

V. Conclusions

The paper introduced two algorithms to support low-delay service migration between edge nodes by leveraging proactive optimization and container-based technologies such as Docker. In particular, we evaluated the proposed algorithms in the openLEON emulator under conditions of limited resources at the edge nodes. The results show that proactive optimization can reduce the service interruption time by around 80% compared to the reactive algorithm.

Armed with the obtained encouraging results, we are now exploring new research directions. On the one hand, are working on mathematical models to manage efficiently the handoff process including the best time to start the handoff. On the other hand, we are assessing the performance of the proposed migration mechanisms in wide-scale scenarios by carrying out application migration experiments over real testbed environments.

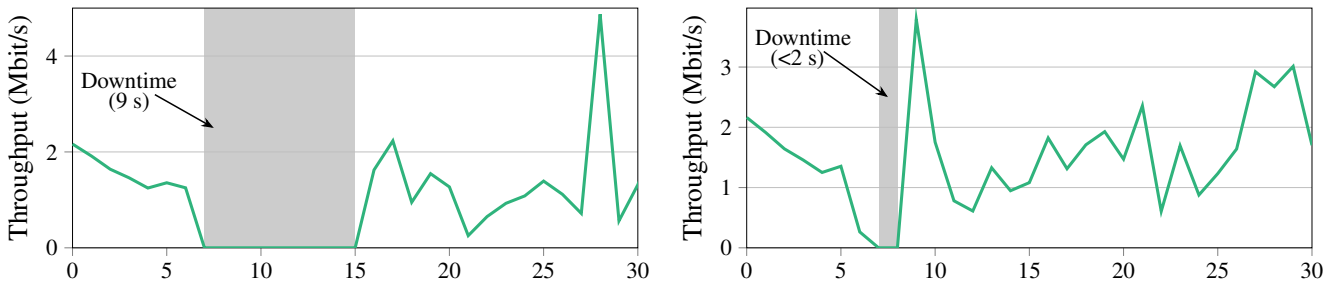
Acknowledgment

Dr. Fiandrino's work is supported by the Juan de la Cierva grant from the Spanish Ministry of Science and Innovation (IJC2019-039885-I).

References

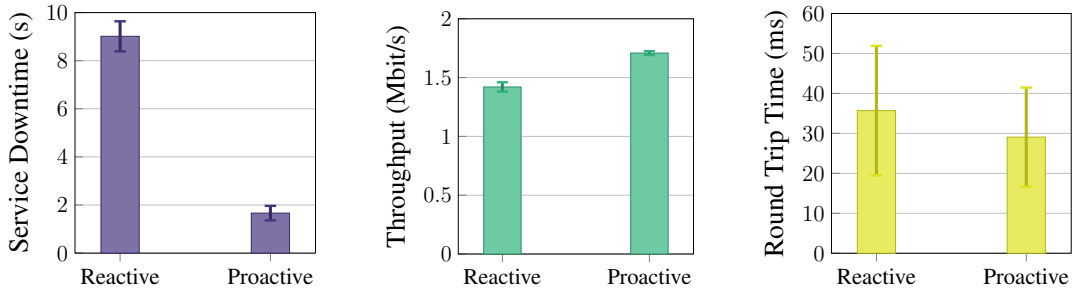
[1] Z. Zhou, X. Chen, E. Li, L. Zeng *et al.*, "Edge intelligence: Paving the last mile of artificial intelligence with Edge computing," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1738–1762, 2019.

[2] T. Qiu, J. Chi, X. Zhou, Z. Ning *et al.*, "Edge computing in industrial Internet of Things: Architecture, advances and challenges," *IEEE Communications Surveys Tutorials*, pp. 1–1, 2020.



(a) Reactive (b) Proactive

Figure 7. Throughput observed during the reactive and proactive handoff



(a) Service downtime (b) Throughput (c) RTT

Figure 8. Performance of reactive and proactive protocols

[3] Y. Mao, C. You, J. Zhang, K. Huang *et al.*, "A survey on mobile edge computing: The communication perspective," *IEEE Communications Surveys Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.

[4] L. Lin, X. Liao, H. Jin, and P. Li, "Computation offloading toward edge computing," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1584–1607, 2019.

[5] T. Taleb, K. Samdanis, B. Mada, H. Flinck *et al.*, "On multi-access edge computing: A survey of the emerging 5G network edge cloud architecture and orchestration," *IEEE Communications Surveys Tutorials*, vol. 19, no. 3, pp. 1657–1681, 2017.

[6] F. Giust, G. Verin, K. Antevski, J. Chou *et al.*, "MEC deployments in 4G and evolution towards 5G," Feb 2018, ETSI White Paper.

[7] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Communications Surveys Tutorials*, vol. 19, no. 3, pp. 1628–1656, Third Quarter 2017.

[8] P. Vitello, A. Capponi, C. Fiandrino, G. Cantelmo *et al.*, "The impact of human mobility on edge data center deployment in urban environments," in *Proc. of IEEE GLOBECOM*, 2019, pp. 1–6.

[9] C. Fiandrino, N. Allio, D. Kliazovich, P. Giaccone *et al.*, "Profiling performance of application partitioning for wearable devices in mobile cloud and fog computing," *IEEE Access*, vol. 7, pp. 12 156–12 166, Jan 2019.

[10] C. Ragona, F. Granelli, C. Fiandrino, D. Kliazovich *et al.*, "Energy-efficient computation offloading for wearable devices and smartphones in mobile cloud computing," in *Proc. of IEEE GLOBECOM*, Dec 2015, pp. 1–6.

[11] T. Taleb, A. Ksentini, and P. A. Frangoudis, "Follow-me cloud: When cloud services follow mobile users," *IEEE Transactions on Cloud Computing*, vol. 7, no. 2, pp. 369–382, 2019.

[12] L. Ma, S. Yi, and Q. Li, "Efficient service handoff across edge servers via docker container migration," in *Proc. of ACM/IEEE SEC*, 2017, pp. 1–13.

[13] L. D. Xu, W. He, and S. Li, "Internet of things in industries: A survey," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 4, pp. 2233–2243, 2014.

[14] R. Zhu, X. Zhang, X. Liu, W. Shu *et al.*, "Erdt: Energy-efficient reliable decision transmission for intelligent cooperative spectrum sensing in industrial iot," *IEEE Access*, vol. 3, pp. 2366–2378, 2015.

[15] C. Fiandrino, D. Kliazovich, P. Bouvry, and A. Y. Zomaya, "Performance and energy efficiency metrics for communication systems of cloud computing data centers," *IEEE Transactions on Cloud Computing*, vol. 5, no. 4, pp. 738–750, Oct 2017.

[16] P. Bellavista, A. Corradi, L. Foschini, and D. Scotece, "Differentiated service/data migration for edge services leveraging container characteristics," *IEEE Access*, vol. 7, pp. 139 746–139 758, 2019.

[17] C. Fiandrino, A. B. Pizarro, P. J. Mateo, C. A. Ramiro *et al.*, "openLEON: An end-to-end emulation platform from the edge data center to the mobile user," *Computer Communications*, vol. 148, pp. 17 – 26, 2019.

[18] C. Dupont, R. Giaffreda, and L. Capra, "Edge computing in IoT context: Horizontal and vertical linux container migration," in *Proc. of GIoTS*, 2017, pp. 1–4.

[19] A. Yousefpour, G. Ishigaki, R. Gour, and J. P. Jue, "On reducing iot service delay via fog offloading," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 998–1010, 2018.

[20] S. Mubeen, P. Nikolaidis, A. Didic, H. Pei-Breivold *et al.*, "Delay mitigation in offloaded cloud controllers in industrial IoT," *IEEE Access*, vol. 5, pp. 4418–4430, 2017.

[21] C. You, K. Huang, H. Chae, and B. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Transactions on Wireless Communications*, vol. 16, no. 3, pp. 1397–1411, 2017.

[22] K. Kaur, S. Garg, G. S. Aujla, N. Kumar *et al.*, "Edge computing in the industrial internet of things environment: Software-defined-networks-based edge-cloud interplay," *IEEE Communications Magazine*, vol. 56, no. 2, pp. 44–51, 2018.

[23] H. Guo and J. Liu, "Collaborative computation offloading for multiaccess edge computing over fiber–wireless networks," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 5, pp. 4514–4526, 2018.

[24] C. Clark, K. Fraser, S. Hand, J. G. Hansen *et al.*, "Live migration of virtual machines," in *Proc. of USENIX NSDI*, 2005, p. 273–286.

[25] A. Machen, S. Wang, K. K. Leung, B. J. Ko *et al.*, "Live service migration in mobile edge clouds," *IEEE Wireless Communications*, vol. 25, no. 1, pp. 140–147, 2018.

[26] S. Nadgowda, S. Suneja, N. Bila, and C. Isci, "Voyager: Complete container state migration," in *Proc. of IEEE ICDCS*, 2017, pp. 2137–2142.

[27] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for vm-based cloudlets in mobile computing," *IEEE Pervasive Computing*, vol. 8, no. 4, pp. 14–23, 2009.

[28] L. Ma, S. Yi, N. Carter, and Q. Li, "Efficient live migration of edge services leveraging container layered storage," *IEEE Transactions on Mobile Computing*, vol. 18, no. 9, pp. 2020–2033, 2019.

[29] M. Peuster, J. Kampmeyer, and H. Karl, "Containernet 2.0: A rapid prototyping platform for hybrid service function chains," in *Proc. of IEEE NetSoft*, 2018, pp. 335–337.