# DINO: Dynamic and Intelligent Network Operative for Secure and Efficient Access to the Edge Network Computing Environment

Caio Storni, Diogo M. F. Mattos and Dianne S. V. Medeiros

***Abstract*— Multi-access Edge Computing (MEC) extends cloud computing to the network's edge, placing MEC servers closer to End Devices (EDs). This paradigm aims to respond locally to computing tasks, reducing the routed data to the core network and the communication latency. Nevertheless, offloaded task requests may congest MEC servers and increase waiting times. Therefore, this paper proposes the Dynamic and Intelligent Network Operative (DINO) MEC server, an edge computing host server that securely and efficiently handles task offloading. The proposal spans the MEC server from the ED premises to the edge network. DINO splits into Customer Premises Equipment (CPE) and Virtualized Infrastructure (VI), and relies on a modular architecture that allows the CPE to be placed anywhere between the ED's premises and the VI running in the edge network. We assess DINO's performance by varying the number of EDs to verify how the modular architecture affects the network overhead and delay. Results show a minimal increase in the experimented delay due to the proposed MEC server, which is outgrown by the delay due to the medium access control protocol. The network overhead is negligible, allowing EDs accessing an edge service to achieve the same data rate as the EDs accessing a cloud service.**

***Keywords*— Multi-access Edge Computing, Network Security, Network Virtualization, Edge Network Computing**

## I. Introduction

Previous generations of data-enabled mobile networks rely on the core network to route data traffic from a data center to a base station close to the mobile End Device (ED) [1]. However, it potentially increases communication latency, mainly due to the propagation delay between the ED and the data center. High-latency communication does not meet the next-generation mobile networks' requirements. Thus, the European Telecommunications Standardization Institute (ETSI) proposes the framework and the reference architecture for Multi-access Edge Computing (MEC) to provide a network solution to deliver services and computing functions to the edge nodes with reliable and ultimate service experience. The MEC architecture allows resources, applications, and services to span the communication path from the ED to the cloud [2]. MEC architecture also supports high-bandwidth and low-latency applications and services bridging cloud computing and end-users [2]. To this end, MEC takes advantage of the existing Network Function Virtualization Infrastructure (NFVI) and NFV Management and Orchestration (MANO). MEC enhances NFVI and MANO with new computing and storage resources and creates a virtualized environment for a wide range of applications running at the mobile network edge [3]. Software-Defined Networking (SDN) and Service Function Chaining (SFC) are two other major building blocks to deploy MEC. SDN separates the control and user planes and logically centralizes the control plane. SFC consistently interconnects multiple virtualized network functions, applying functions in a specific order suitable to the MEC environment. SDN and SFC/NFV allow the implementation of network slicing to achieve performance requirements for each MEC application by dynamically setting up logical networks.

Several challenges exist when deploying a MEC environment. For instance, the high number of EDs' network flows may overload the SDN controller and degrade the network performance and Quality of Service (QoS) guarantees. Hence, placement of the MEC components throughout the network environment is a complex task and essential to ensure that the control and data planes at the edge network meet the requirements of the constantly oscillating network flows of EDs. In this context, there is a need for solutions that support efficient tasking offloading schemes in MEC-enabled environments. This paper proposes the Dynamic and Intelligent Network Operative (DINO) MEC Server, an edge computing host server that spans from the ED premises to the edge network. To deploy DINO MECS, we propose a network architecture that relies on traditional TCP/IP protocols. We split the server into Customer Premises Equipment (CPE) and a Virtualized Infrastructure (VI), connected through a secure virtual communication link. We aim to reduce the overall latency to access MEC applications. We assess the performance of the DINO MECS' proposed network architecture in different scenarios, varying the number of connected clients and the traffic load. The results show that the delay due to the medium access control protocol outshines the delay due to the proposed network architecture. Hence, the added delay is minimal. Moreover, our architecture does not influence network fairness, and the network overhead is negligible.

The remainder of this paper is organized as follows. Section II briefly discusses related work. Section III reviews the Multi-access Edge Computing system according to ETSI. Section IV presents the DINO MECS model and the proposed underlying network architecture, discussing the implementation. Section V presents the results. Finally, Section VI concludes this paper and provides future work directions.

Caio Storni, Diogo Mattos and Dianne Medeiros, Departamento de Engenharia de Telecomunicações, Universidade Federal Fluminense, Niterói-RJ, e-mail: {stornicaio,diogo_mattos,diannescherly}@id.uff.br. This work is partially funded by RNP, CAPES, FAPERJ, CNPq, and Niterói City Hall/Fundação Euclides da Cunha/Universidade Federal Fluminense.

## II. Related Work

Mattos *et al.* propose a distributed SDN controller architecture, not fastened to any specific network environment, aiming to maintain a global view of the network by establishing independent areas [4]. Li et al. [5] investigate the SDN controller placement problem modeled into a multi-objective optimization problem, considering an SDN-based Internet of Vehicles (IoV) environment using Deep Reinforcement Learning (DRL). The authors propose novel models for jointly achieving optimal latency, load balance, and high reliability in a dynamic network environment based on an edge controller, a domain controller, and a root controller in a three-layer architecture. Similarly, security remains a significant challenge, as MEC joins various building blocks to enable the technologies and techniques for computation offloading [2]. As a result, MEC introduces risks related to all its building blocks. Moreover, the MEC Servers (MECSs) have limited computing and storage capacity, posing a challenge to serving many users, as resources can be depleted quickly. Congestion on the MECS due to frequent task offloading from EDs ceases the advantages of deploying a MECS, as more prolonged waiting times override the provided ultra-low latency.

Bolettieri et al. [6] focus on network slicing, designing a multi-tenant MEC architecture for application and network slicing, considering different operational and business roles. The authors neglect security issues related to communication within the proposed architecture. Lee et al. [7] propose a mobile personal MEC architecture using the user's mobile device as MECS. The goal is to provide faster responses and deliver services continuously to the end user. The proposed scheme reduces the average service delay and provides efficient task offloading compared to the existing MEC scheme due to using mobile user devices as MECS. The approach, however, increases the consumption of users' device computational power, which may lead to quicker battery depletion due to increased energy consumption. These works do not focus on secure communication with the MECS nor the deployment of MECS on the ED premises without affecting the ED performance. Most works related to MEC architectures do not address fundamental technical issues in the architecture design, such as partitioning the functions between the network and application layers. Furthermore, a MEC architecture must consider computing and communication models that do not modify the traditional TCP/IP protocol stack [8].

Differently, we aim to support efficient task offloading by deploying a MECS as an edge computing host server in the MEC environment without affecting user devices' energy consumption while focusing on the easy deployment of such servers. Our solution is compatible with the traditional TCP/IP protocol stack to facilitate integration with the current network and to be agnostic to the network environment.

## III. Multi-access Edge Computing (MEC)

The ETSI GS MEC 003 V3.1.1. (2022-03) describes the MEC system that enables MEC applications to run efficiently and seamlessly in a multi-access network [9]. The MEC generic reference architecture defines a MEC host level and a MEC system level [2], [9], [10]. At the host level, MEC architecture contains the MEC host, the MEC platform manager, and the Virtualization Infrastructure Manager (VIM). The *MEC host* comprises the MEC platform, MEC applications, and Virtualization Infrastructure (VI). The VI provides computing, storage, and network resources for MEC application instantiation. The VI's data plane executes traffic rules received by the MEC platform and routes traffic among applications, services, DNS server/proxy, access networks, local networks, and external networks. The MEC platform is a set of paramount functionality required to run the MEC applications on the VI, enabling them to provide and consume MEC services. The MEC platform receives traffic rules from the MEC platform manager, applications, or services and instructs the data plane accordingly. The MEC applications are instantiated on the VI and can interact with the MEC platform to consume and provide MEC services. The *MEC platform manager* is responsible for keeping the applications' lifecycle, providing element management functions to the MEC platform, and supervising the application rules and requirements. The *VIM* has several responsibilities, such as managing the VI's computing, storage, and networking resources and preparing the VI to run a software image. The VIM also reallocates applications from and to external cloud environments according to the applications requirements [9].

The MEC system level contains the MEC Orchestrator (MEO), the Operations Support System (OSS), the device application, and the Customer Facing Service (CFS) portal. The *MEO* is the core functionality in the MEC system-level management. It maintains a global view of the MEC system based on deployed MEC hosts, available resources, MEC services, and topology. The MEO selects appropriate MEC hosts for instantiating and relocating applications and terminates applications. The *OSS* receives requests via the CFS portal and from device applications for instantiation or termination. The interactions happen via a user-application lifecycle-management proxy, and the granted requests are forwarded from the OSS to the MEO for further processing. The *CFS portal* allows third-party customers to select and order a set of MEC applications and to receive back service level information from the provisioned applications.

The MEC architecture also structs as layers according to functional elements. The structure comprises EDs at the bottom layer, followed by the access network, the edge network, and the core network layers. The end devices layer includes all EDs connected to the access network, such as User Equipment (UEs) and Internet of Things (IoT) devices. The access network layer comprises the networks connecting EDs to the edge or core networks. The edge network layer contains the edge network infrastructure owned by an infrastructure provider owns it. The MEC deployment may occur through multiple edge networks that continuously cooperate and remain connected to the traditional cloud [11]. The core infrastructure resides in the core network layer, where the centralized MEC control and management functions for the EDs are deployed [2]. Communication between EDs and services can happen directly with cloud servers on the Internet through the core network layer or with MEC Servers (MECS) through

the edge network layer. MECS are edge servers that exist along with the VI [2] in the edge network layer. The MECSs are small-scale data centers that accommodate the MEC host and provide the computing, storage, and network capabilities that cloud data centers once provided centralized. The provided capabilities are shared by multiple virtual machines running on top of these servers [3]. MECSs are managed and owned by the infrastructure provider and can be deployed anywhere between the Radio Access Network (RAN) and the edge of the core network [11]. Hence, offloading tasks to the MECS reduces the latency compared to offloading to a cloud server.

## IV. The DINO MEC Server Model

We propose DINO MECS, an operative that supports secure connections from different applications and enables computing and storing capabilities at the edge network, reducing task load on EDs and providing low latency. DINO MECS supports efficient handling of task offloading in a MEC environment by combining hardware resources and a virtual layer managed by the DINO module. We deploy a MEC-based architecture that allows EDs to connect to DINO MECS. We assume that part of the MECS hardware is a piece of Customer Premises Equipment (CPE) co-located with the Access Point (AP) at the ED premises. The other part of the MECS hardware resides in a Virtualized Infrastructure (VI). The separation is advantageous, as the VI forms an edge cloud that may be placed anywhere within the edge network. The CPE provides EDs with automatic and secure access to the VI. Hence, DINO MECS seamlessly spans from the ED premises to the edge network, agnostic to the underlying access network.

The DINO MECS follows a modular architecture, shown in Figure 1, that allows the simple and practical deployment of new functionalities. The main module is the DINO module, divided into a client- and a server-side component. The module manages communication between the ED premises' CPE and the edge network VI. The CPE provides access to EDs, creating managed Wireless Local Area Networks (WLANs). The software suite embedded into the CPE divides into (i) DINO Client (DINO-C), (ii) Remote Connection Client (RCC), and (iii) Network Virtualization (NV) modules. The (i) DINO-C module manages the network connection with the VI and enforces actions received from the DINO Server (DINO-S). The (ii) RCC creates two virtual private connections, one for control and the other for data. The control connection reaches a virtualized firewall in the VI provided by a Network Security module (NS), while the data connection reaches a target Virtual Machine (VM) behind the firewall. The VM runs, e.g., MEC applications that provide services to EDs. A Virtual Private Network (VPN) software establishes both connections. We also deploy a Generic Routing Encapsulation (GRE) tunnel in the data VPN to allow any traffic to be transmitted as a MAC-based link-layer network. As such, the underlying network infrastructure becomes transparent. We use OpenVPN[1] to establish the VPNs. The (iii) NV module allows the creation of virtualized network interfaces over the CPE's physical network interface. To this end, we use Open vSwitch (OVS)[2], a multilayer software switch that provides a switching stack for hardware virtualization and allows the creation of OpenFlow-enabled virtual switches. Hence, the CPE generates managed WLANs and correctly forwards the traffic to the edge network or the Internet. If the user accesses a MEC service, traffic flows to the VI. Otherwise, the traffic routes to the Internet via the carrier's core network.

Figure 2 shows the network communication scheme to enlighten how the CPE correctly forwards the traffic. The DINO-C generates an "Edge WLAN" and a "Cloud WLAN" using two bridges, `wlan edge` and `wlan cloud`. The NV module creates two virtual network interfaces connecting to a virtual switch provided by Open vSwtich (OVS). OVS connects to the CPE's physical Ethernet interface (`eth0`). The DINO-C attaches a GRE tunnel to the OVS for carrying traffic from the Edge WLAN, aiming to reach applications and services behind the virtualized firewall at the IV. The Cloud WLAN is logically attached to a `local` interface, allowing traffic to flow directly to the Internet. Therefore, EDs in the Edge WLAN connect to the edge network, whereas EDs in the Cloud WLAN connect to the Internet. The data VPN encapsulates the GRE tunnel and is established with the MEC tenant VM. The control VPN is established with the DINO Server (DINO-S) module to manage communication and uses TCP as the transport protocol. DINO-S also uses a virtual router to forward traffic for tenants in the VI correctly [12].

The VI software suite divides into (i) DINO Server (DINO-S), (ii) Network Security (NS), (iii) Network Controller (NC), (iv) Remote Connection Server (RCS), (v) Identity and Access Manager (IAM) and (vi) Virtualization Infrastructure Manager (VIM) modules. The (i) DINO-S module manages the network connection with the CPE, serving as an interface between the CPE and the NS, NC, RCS, and IAM modules. The DINO-S module also provides an interface with any new service or application running on the VI. The (ii) NS module inspects packets and enforces the security policy rules to block unapproved communication with the VI. We deploy Pfsense[3] as the NS, acting as a virtual router and enabling features such as firewall, intrusion detection, and malware detection [13]. The (iii) NC is responsible for managing the network flow. We use Open Network Operating System (ONOS)[4] as NC. The (iv) RCS receives the remote connections from the RCC, establishing a secure channel for control and data traffic. The (v) IAM provides authentication, authorization, and accountability, allowing identity and policy management. We deploy as IAM the Free Identity, Policy and Audit (FreeIPA)[5] software, which is Free and Open Source (FOSS) software. Finally, the (vi) VIM module is primarily responsible for managing application and service provisioning using the resources at the edge network's VI and maintaining information about these resources, such as topology, available resources, and services. We deploy as VIM the Openstack[6] platform, which is open source and provides cloud infrastructure for VM, bare metal,

[1] Available at https://openvpn.net/.

[2] Available at https://www.openvswitch.org/.

[3] Available at https://www.pfsense.org/.

[4] Available at https://opennetworking.org/onos/.

[5] Available at https://www.freeipa.org/.

[6] Available https://www.openstack.org/

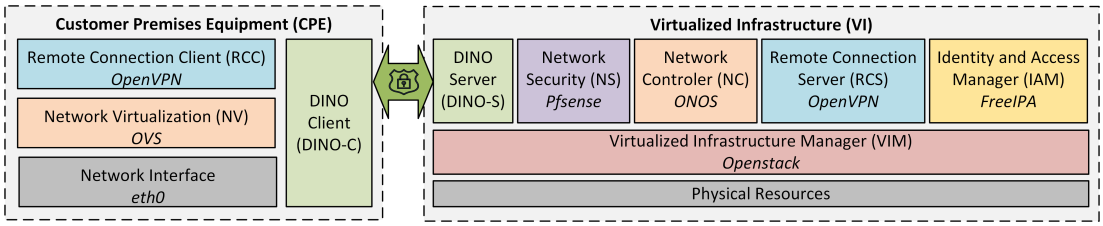



Fig. 1. DINO MECS spans from the UE premises to the edge network, and it is composed of a software suite divided into CPE and VI. The DINO module manages the communication between the CPE and VI through a secure connection.

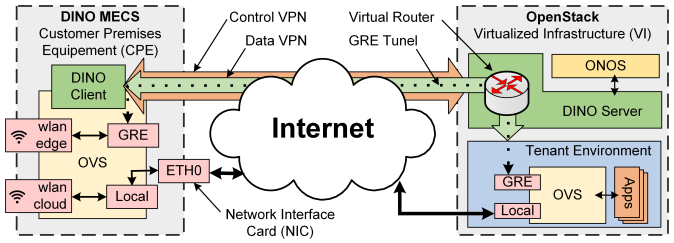
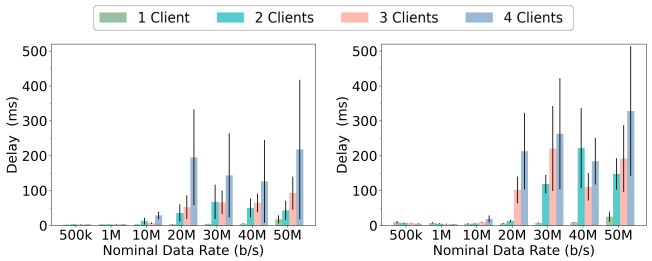
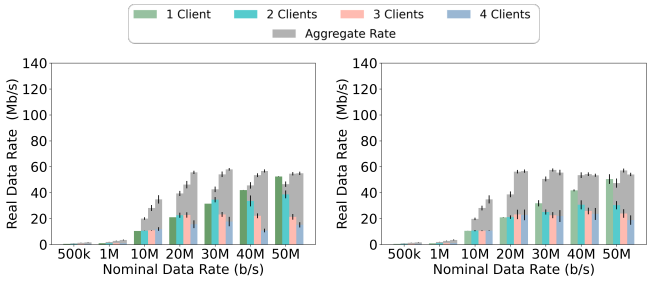



Fig. 2. DINO MECS is a piece of CPE that connects to the Edge Network through a couple of VPNs and a GRE tunnel. The VI deployment is over OpenStack platform. The centralized DINO-S deploys an ONOS SDN controller to control the operation of DINO MECS instances. As a result, each tenant experiences an isolated edge network environment.




(a) Local Setup. (b) Edge Setup.

Fig. 3. Delay experienced by clients when the NDR and the number of clients increase. The experimented delay is mainly due to the medium access control protocol in the WLAN.




(a) Local Setup. (b) Edge Setup.

Fig. 4. Real data rate clients achieve when the NDR and the number of clients increase. Our architecture does not influence network fairness, as the connected clients receive equal fractions of the network bandwidth in both the Edge and Local Setups.

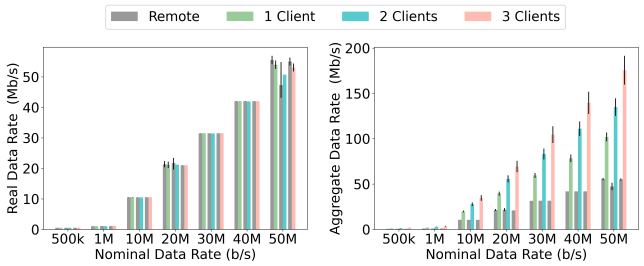
and containers. Hence, Openstack controls the computing, storage, and networking resources pool.

The MEC environment joins cloud computing and network virtualization, inheriting threats from both domains. MEC-based solutions must tackle entity authentication, identity verification, network security with traffic separation, application integrity assurance, malware detection within the MEC layer, data encryption, and temper-proof MEC equipment [2]. Our proposal addresses these requirements, excepting application integrity assurance and temper-proof MEC equipment, which are not in the scope of this work. The IAM and RCC/RCS modules address entity authentication and identity verification, while the NV and NC modules tackle network security with traffic separation, aided by the RCC/RCS modules. The NS module addresses malware detection within the MEC layer, and the RCC/RCS modules address data encryption.

## V. Results and Discussion

We assess the performance of the proposed DINO MECS network architecture in different scenarios, varying the number of connected clients and the traffic load. The main goal is verifying the added overhead's impact on communication. To this end, we deploy a testbed composed of four clients, one DINO CPE at the local premises and the DINO VI at the edge network. The clients play the role of edge devices. Clients and the DINO CPE are Raspberry PI 4 Model B boards with 2GB RAM, running Ubuntu 20.04. Clients run `iperf` to generate a Constant Bit Rate (CBR) traffic within the Nominal Data Rate (NDR) set $\{0.5, 1, 10, 20, 30, 40, 50\}$ Mb/s. The traffic directly targets DINO VI (Edge Setup) or DINO CPE (Local Setup). We evaluate the local network in the Local Setup, and our architecture has no influence. In turn, the Edge Setup assesses the overhead of remote connections and the GRE tunnel by sending data to an application VM in the DINO VI. The testbed's edge network is a private cloud deployment at Universidade Federal Fluminense (UFF) in Niterói/RJ, Brazil.

In the first scenario, we evaluate the delay clients experience in each setup. We run ten `ping` tests varying the number of connected clients for each data rate. Figure 3 shows the results with a confidence interval of 95%. The delay varies more when the number of clients or traffic increases in all setups. Figure 3(a) shows that the delay in the Local Setup increases with the growth of the sent traffic, and, within the same data rate, the delay increases with the number of clients. It is an expected result due to the contending over access to the transmission medium. Hence, as clients are subject to the CSMA/CA protocol, they experiment with increasing and varying delays due to risen time to access the medium. When traffic targets the application VM and, thus, flows through the GRE tunnel and the data and control VPNs, the delay behavior is similar to the Local Setup, as shown in Figure 3(b). However, due to the high variation of the delay in both setups for higher data transmission rates and the number of clients, it is not directly correlated whether the proposed




(a) Real data rate. (b) Aggregated data rate.

Fig. 5. Real data rate achieved by clients connected on the Edge and Cloud WLANs when the NDR and the number of connected clients increase. One client connects to the Edge WLAN, while up to three clients connect to the Cloud WLAN. Increasing the number of clients in the Cloud WLAN does not affect the client connected to the Edge WLAN.

modular architecture significantly influences the delay. Nevertheless, considering one client and any NDR, the influence of CSMA/CA over the experimented delay is negligible, and we observe that the delay is statistically the same for both setups. Therefore, we infer that the experimented delay due to the proposed modular architecture is minimal.

In the second scenario, we capture the traffic generated by `iperf` to evaluate the real data rate clients achieve when the NDR and the number of clients increase. Figure 4 shows the results for each setup. The aggregated data rate represents the sum of all clients' real data rates. Figures 4(a) and 4(b) show that the real data rates for the Local and Edge Setups have similar behavior. The connected clients receive equal fractions of the network bandwidth if the aggregated data rate is lower than 54 Mb/s, which is the upper-bound data rate due to limitations on the wireless network interfaces. Hence, the proposed MEC architecture does not hamper network fairness.

In the last scenario, we evaluate whether increasing the data traffic and the number of clients connected to the Cloud WLAN influence the real data rate achieved by the client connected to the Edge WLAN. In this scenario, DINO CPE deploys an additional wireless network card, a Realtek 8821CU Wireless LAN 802.11ac USB. The additional network card deploys a 2.4 GHz wireless network for the Cloud WLAN. We capture the CBR traffic generated by `iperf` on both wireless networks (Cloud and Edge WLANs). Figure 5 shows the results considering one client connected to the Edge WLAN, namely edge client, and up to 3 clients connected to the Cloud WLAN, namely cloud clients. The edge client achieves a real data rate equal, or statistically close, to the nominal data rate, independently of the sent data rate and the number of cloud clients, as shown in Figure 5(a). The cloud clients also achieve the NDR. Figure 5(b) shows that the aggregated data rate for the edge client is equal to the NDR, as it is the only client in the Edge WLAN. Focusing on the Cloud WLAN, the aggregated data rate increases when the NDR and the number of cloud clients increase, as expected. Hence, the traffic load on the Cloud WLAN does not affect the actual data rate achieved by the edge client.

## VI. Conclusion

Edge computing is present in everyday life through end devices, such as smartphones or Internet of Things devices in local networks. However, accessing the edge network is still a challenge. Thus, this paper proposed the Dynamic and Intelligent Network Operative (DINO) for Multi-access Edge Computing. DINO's goal is to securely extend the Edge Network to the local access network in a transparent, practical, and cost-effective manner. The paper evaluated the proposed implementation. The results show that the proposal introduces a negligible delay in communication between end devices and the edge network, maintains fairness in sharing networking resources, and allows end devices to access the total capacity of the wireless access link. Moreover, we also show that the local wireless network and the edge network accesses are isolated from the performance standpoint. We envision removing in future work the need for two separate WLANs by developing an automatic mechanism to split the traffic correctly to the cloud and to the edge.

## References

[1] M. A. Lopez, G. N. N. Barbosa, and D. M. F. Mattos, "New barriers on 6g networking: An exploratory study on the security, privacy and opportunities for aerial networks," in *2022 1st International Conference on 6G Networking (6GNet)*, 2022, pp. 1–6.

[2] B. Ali, M. A. Gregory, and S. Li, "Multi-access edge computing architecture, data security and privacy: A review," *IEEE Access*, vol. 9, pp. 18 706–18 721, 2021.

[3] B. Blanco, J. O. Fajardo, I. Giannoulakis, E. Kafetzakis, S. Peng, J. Pérez-Romero, I. Trajkovska, P. S. Khodashenas, L. Goratti, M. Paolino, E. Sfakianakis, F. Liberal, and G. Xilouris, "Technology pillars in the architecture of future 5G mobile networks: NFV, MEC and SDN," *Computer Standards & Interfaces*, vol. 54, pp. 216–228, 2017, sI: Standardization SDN&NFV.

[4] D. M. F. Mattos, O. C. M. B. Duarte, and G. Pujolle, "A resilient distributed controller for software defined networking," in *2016 IEEE International Conference on Communications (ICC)*, 2016, pp. 1–6.

[5] B. Li, X. Deng, X. Chen, Y. Deng, and J. Yin, "MEC-based dynamic controller placement in SD-IoV: A deep reinforcement learning approach," *IEEE Transactions on Vehicular Technology*, vol. xx, no. preprint, pp. 1–15, 2022.

[6] S. Bolettieri, D. T. Bui, and R. Bruno, "Towards end-to-end application slicing in multi-access edge computing systems: Architecture discussion and proof-of-concept," *Future Generation Computer Systems*, vol. 136, pp. 110–127, 2022.

[7] J. Lee, J.-W. Kim, and J. Lee, "Mobile personal multi-access edge computing architecture composed of individual user devices," *Applied Sciences*, vol. 10, no. 13, 2020.

[8] H. Liu, F. Eldarrat, H. Alqahtani, A. Reznik, X. de Foy, and Y. Zhang, "Mobile edge cloud system: Architectures, challenges, and approaches," *IEEE Systems Journal*, vol. 12, no. 3, pp. 2495–2508, 2018.

[9] Multi-access Edge Computing (MEC) ETSI Industry Specification Group (ISG), "Multi-access Edge Computing (MEC): Framework and reference architecture," European Telecommunications Standards Institute, Tech. Rep. ETSI GS MEC 003 V3.1.1 (2022-03), Mar. 2022.

[10] P. Ranaweera, A. D. Jurcut, and M. Liyanage, "Realizing multi-access edge computing feasibility: Security perspective," in *2019 IEEE Conference on Standards for Communications and Networking (CSCN)*, 2019, pp. 1–7.

[11] N. Slamnik-Kriještorac, M. Peeters, S. Latré, and J. M. Marquez-Barja, "Analyzing the impact of VIM systems over the MEC management and orchestration in vehicular communications," in *29th International Conference on Computer Communications and Networks (ICCCN 2020)*, vol. 29, no. 1, 2020, pp. 1–6.

[12] D. M. F. Mattos, L. H. G. Ferraz, L. H. M. K. Costa, and O. C. M. B. Duarte, "Evaluating virtual router performance for a pluralist future internet," in *Proceedings of the 3rd International Conference on Information and Communication Systems*, ser. ICICS '12. New York, NY, USA: Association for Computing Machinery, 2012. [Online]. Available: https://doi.org/10.1145/2222444.2222448

[13] M. Andreoni Lopez, D. M. F. Mattos, O. C. M. B. Duarte, and G. Pujolle, "A fast unsupervised preprocessing method for network monitoring," *Annals of Telecommunications*, vol. 74, no. 3, pp. 139–155, Apr 2019. [Online]. Available: https://doi.org/10.1007/s12243-018-0663-2