Journal of Parallel and Distributed Computing 150 (2021) 155–183

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

journal homepage: www.elsevier.com/locate/jpdc

A review of edge computing: Features and resource virtualization

Yaser Mansouri [*], M. Ali Babar

Centre for Research on Engineering Software Technologies (CREST), School of Computer Science, The University of Adelaide, Adelaide, Australia

A R T I C L E   I N F O

Article history:
Received 5 August 2020
Received in revised form 7 November 2020
Accepted 20 December 2020
Available online 6 January 2021

Keywords:
Mobile Edge Computing
Fog Computing
Classical virtualization
Network function Virtualization (NFV)

A B S T R A C T

With the advent of Internet of Things (IoT) connecting billions of mobile and stationary devices to serve real-time applications, cloud computing paradigms face some significant challenges such as high latency and jitter, non-supportive location-awareness and mobility, and non-adaptive communication types. To address these challenges, edge computing paradigms, namely Fog Computing (FC), Mobile Edge Computing (MEC) and Cloudlet, have emerged to shift the digital services from centralized cloud computing to computing at edges. In this article, we analyze cloud and edge computing paradigms from features and pillars perspectives to identify the key motivators of the transitions from one type of virtualized computing paradigm to another one. We then focus on computing and network virtualization techniques as the essence of all these paradigms, and delineate why *virtualization features*, *resource richness* and *application requirements* are the primary factors for the selection of virtualization types in IoT frameworks. Based on these features, we compare the state-of-the-art research studies in the IoT domain. We finally investigate the deployment of virtualized computing and networking resources from performance perspective in an edge-cloud environment, followed by mapping of the existing work to the provided taxonomy for this research domain. The lessons from the reviewed are that the selection of virtualization technique, placement and migration of virtualized resources rely on the requirements of IoT services (i.e., latency, scalability, mobility, multi-tenancy, privacy, and security). As a result, there is a need for prioritizing the requirements, integrating different virtualization techniques, and exploiting a hierarchical edge-cloud architecture.

1. Introduction

Cloud computing is the delivery of centralized and virtualized computing, storage, services, and application resources over the Internet. Cloud Computing decouples services from underlying infrastructure and eliminates upfront cost and complexity of managing IT infrastructure [42]. However, cloud computing is not able to serve real-time Internet of Things(IoT) applications because centralized cloud infrastructure is usually located at a few fixed places, which are away from users; moreover, IoT applications require *ultra-low latency*, *low jitter*, *high-demand bandwidth*, *mobility services*, to name a few [189].

The above-mentioned requirements of IoT applications can be met to some extent through paradigms that are fundamentally created based on a cloud computing infrastructure. Spanning Cloud Computing (SCC) [141,229] is a cloud-based delivery model in which an application requiring a large pool of computing resources is deployed over multiple cloud datacenters to achieve better performance, higher availability and lower monetary cost [67,68,139]. SCC is a double-edged sword cloud-based paradigm since it imposes several overheads such as management and orchestration of resources and requires a security mechanism to protect decentralized resources across different cloud datacenters. Cloud-based Content Delivery Network (CCDN) [242] is another cloud-based paradigm that takes the advantage of Geo-distributed and pay-as-you-go model of cloud platforms and provides content-delivery-as-a-service. Similar to the traditional CDNs, CCDNs such as Amazon CloudFront [16], Google CDN [88] and Azure CDN [23] enhance Quality of Experience (QoE) through delivery of content by replicating them in the vicinity of the content requester [201,272]. Although both SCC and CCDNs move data processing and storage closer to users and improve performance metrics, they are far behind the requirements of IoT applications.

To meet the requirements of IoT applications, there is a need for cloud-based services in proximity of data source (e.g., IoT devices/applications) to process, analyze and filter data for taking proper actions. The extension of computing services from centralized cloud-based paradigms to the edge of network is called *edge computing* [113,199,263] that boosts the overall efficiency of infrastructures by achieving *ultra-low latency*, *reducing backhaul load*, *supporting mobility services*, and *increasing service resilience*.

Edge computing paradigms consist of connected resource-constrained devices such as smartphones, wearable gadgets,

* Corresponding author.
E-mail address: yaser.mansouri@adelaide.edu.au (Y. Mansouri).

https://doi.org/10.1016/j.jpdc.2020.12.015
0743-7315/© 2020 Elsevier Inc. All rights reserved.

single-board computers (SBCs), network devices (switches, routers) and resources with moderate capabilities such as base stations (BSs), LANs, Cloudlets [204], to name a few [144]. Devices generate and collect a huge amount of data that is pre-processed and analyzed initially at the edge of a network. This data can then be transferred to a centralized cloud for extracting deep knowledge using different Machine Learning (ML) and Optimization (convex or Bayesian) approaches [149,222]. Thus, edge computing is complementary to cloud computing. Mobile Edge Computing (MEC) [1,6,143] and Fog Computing (FC) [162,165] are the most prominent paradigms of edge computing to harness the power of distributed edge resources to support IoT applications such as smart houses, grids, retails, farming, connected cars and healthcare.

Like cloud computing, the fundamental technology of edge computing paradigms is resource virtualization that decouples hardware resources from software in order to run multiple tenants on the same hardware [12,39,157,158]. Nevertheless, edge computing has distinctive differences with cloud computing in terms of *location awareness*, *mobility-based services*, *resource-constrained and heterogeneous devices* and *wide-spread distribution of devices*. These differences imply that heavyweight virtualization (running virtual machine monitor (VMM) directly on either hardware or OS) [85,233] cannot be applicable to all use cases of IoT applications that may leverage edge computing. This constraint leads to the deployment of lightweight virtualization techniques such as containers [208] and unikernels [135] at the edge devices with limited and moderate resource capabilities. These techniques can be scaled down to edge resources as they require less CPU cycles, smaller memory footprint size, less instantiation time and more agility in mobility. The pure usage of one technique may not be performance wise effective [159, 160] and a combination of techniques can be utilized based on the *virtualization features*, *resource capabilities* and *application requirements*.

In addition to the virtualization of computing resources, there is a strong tendency to virtualize networking resources to meet the requirements of IoT applications and to cope with the dynamism of a network status with the help of Software Defined Network (SDN) framework [117]. Network Function Virtualization (NFV) [168] decouples service/network functions (SFs/NFs) such as firewalls, load balancers, Intrusion Detection system (IDS) from underlying network hardware and converts them into Virtualized Network Functions (VNFs). VNFs run on classical VMs, containers and unikernels to execute Service Function Chains (SFCs) with direct and causal dependencies. Due to the existence of such dependencies, it is essential to design algorithms for NFV placement and migration at the edge-cloud environments. The proposed algorithms for classical VMs are not directly applicable since the classical VMs run users' requests without any dependencies.

Related Surveys: With respect to the management of computing and network resources in an edge-cloud environment, researchers have surveyed and analyzed different aspects of FC [261], MEC [72,133] and cloudlet [204] from definitions, architecture, resource management and security perspectives. Some studies briefly discuss the concept of different edge computing paradigms, compare them based on several features and identify their roles in IoT applications [184,234,259,259]. They mostly focus on the requirements and configuration of edge devices, which make them suitable to large scale IoT applications.

Several researchers have conducted surveys on architecture and resource management in edge computing. Hong et al. [100] discussed infrastructures, architectures and algorithms for resource discovery and management in FC. Yousefpour et al. [261] have delineated different edge computing paradigms in details from definition and usage perspective. The authors briefly have discussed FC from different aspects of performance and security. Another study [7] has provided an exhaustive details of Service Placement Problem (SPP) in the context of FC. Similarly, Mouradian et al. [162] discussed applications specific architectures and algorithms for resource sharing, task scheduling and load balancing in FC. Naha et al. [166] studied different architectures of FC in details and review resource allocation and scheduling, fault tolerance and simulation tools in FC environments. Some researchers have analyzed different aspects of security in edge computing paradigms. A recent survey in [199] focused on a detailed analysis of different threat models that impede the integrity of edge computing paradigms. The authors in [170] delineated security and privacy challenges in FC and analyzed promising solutions to tackle these challenges. In another study [114], the authors argued security challenges of applications and potential solutions in the FC context.

There are several valuable survey papers on MEC, as a subset of FC, from different aspects. Mach et al. [133] have reviewed different architectures of MEC and substantially focused on computation offloading from user's perspective. Abbas et al. [1] have presented a survey on MEC based on an investigation of the related technologies. They discussed deployment of different applications in the MEC context from performance and security perspectives. Mao et al. [143] have provided an extensive survey with focal point on joint radio-and-computational resource management. Wang et al. [248] have presented a survey on MEC that covers the convergence of computing, caching, and communication techniques in MEC. The authors in [245] have surveyed research efforts on service migration in MEC and briefly summarized the three hosting technologies (virtual machine, container and agent) for mobile applications.

Several survey studies in the NFV context have been conducted in terms of architectures, management and security. Bonfim et al. [39] have provided a systematic literature review on the integrated architecture of SDN/NFV and NFV deployment scenarios. Mijumbi et al. [150] have explained the relationship between NFV, SDN, and cloud infrastructures, NFV standards and currently implemented NFV projects. In [253], the authors have analyzed different strategies of NFV placement as well as their cons and pros. A comprehensive survey paper [260] covered many NFV aspect varying form NFV architectures, NFV standards, to different algorithms for NFV placement, scheduling and migration. The authors in [122,257] have debated security threats in the NFV infrastructures and the existing solutions for the identified threats. Table 1 summarizes a comparison between our work and the state-of-the-art efforts in the context of virtualization at the edge computing.

Paper contribution and organization: Although the above-discussed survey papers are inspiring, none of them specifically investigates the research issues in virtualization of computing and networking resources at the edge network. Also neither do they discuss to leverage virtualization to support micro-services in IoT frameworks to meet the requirements of IoT applications. Although these studies indicated reasons to leverage edge computing, nor do they present the reason behind shifting from cloud computing and its derivates (i.e., SCC and CCDN) to the edge computing paradigm from more than 20 features and 4 pillars. Motivated by these research gaps, we aimed to cover four specific subjects. (1) Investigating the reason behind moving from cloud computing paradigms (CC, SCC, and CCDN) to the edge computing paradigms (MEC and FC). (2) Discussing different virtualization techniques and their capability to deploy in IoT frameworks, and mapping the state-of-the-art studies with respect to the selected factors of virtualization techniques. (3) Delineating placement and migration of virtualized computing

Table 1
Contribution of the survey papers on virtualization of the cloud-edge infrastructure.

Work	Context	Computing Virt.	Networking Virt.	Objectives
Salaht et al. [200]	Fog and edge (service placement problem)	No	No	Investigating optimization algorithms for services placement
Hong and Varghese [100]	Fog/edge computing (resource management)	No	No	Investigating architectures, infrastructure, and algorithms for managing resources
Ren et al. [196]	All edge paradigms (Architecture and characteristics)	Partial	Partial	Discussing offloading computation, caching, security, and privacy
Mouradian et al. [162]	Fog (Architecture and algorithm)	No	No	Discussing application-agnostic/specific architecture and algorithms for resource sharing, task scheduling, offloading, load balancing
Dolui and Datta [62]	All edge paradigms (Implementation)	No	No	Implementation of different fog computing
Khan et al. [114]	Fog Computing (security)	No	No	Investigating security issues and possible solutions in fog computing
Naha et al. [165]	Fog (Architecture and application requirements)	No	No	Discussing the requirements of infrastructure, platform, and applications
Capra et al. [44]	Edge computing (hardware requirements)	No	No	Discussing hardware requirements for IoT
Roman et al. [199]	Mobile edge (security)	No	No	Discussing threats and security issues in FC, MEC, and MCC
Yi et al. [259]	Fog (Concepts)	No	No	Discussing essential concepts in terms of virtualization, security and privacy, QoS, computation offloading, resource management, and so on
Ni et al. [170]	Fog (Architecture, concepts, security)	No	No	Discussing architecture and role of fog computing in IoT frameworks as well as its security and privacy issues
Abbas et al. [1]	MEC (architecture and application)	No	No	Presenting different architectures and applications deployment in the MEC context
Mach et al. [133]	MEC (Architecture and computation offloading)	Partial	Partial	Discussing different MEC architecture and computation offloading in MEC servers
Mao et al. [143]	MEC (communication)	No	No	Providing a focus on joint radio and communication resource management, and discussing issues in terms of mobility, energy, caching, mobility, and privacy in the context of MEC servers
Beck et al. [29]	MEC (applications deployment)	No	No	Analysing opportunities and limitation of applications deployment from technical perspective in MEC servers
Wang et al. [245]	MEC (service migration)	Partial	No	Discussing live migration for datacenters and handover in cellular networks
Bonfim et al. [39]	Edge (architecture)	No	Partial	Investigating in-depth review of NFV/SDN architectures and synthesizing their design
Our Work	Edge (virtualization)	Yes	Yes	Investigating virtualization of computing and networking resources in edge computing from features and performance perspective

Virt: virtualization.

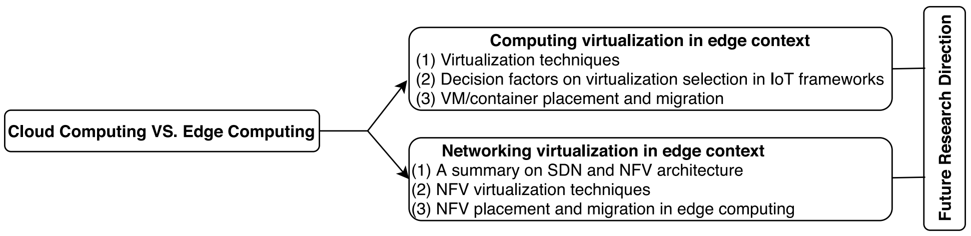

Cloud Computing VS. Edge Computing
Computing virtualization in edge context
(1) Virtualization techniques
(2) Decision factors on virtualization selection in IoT frameworks
(3) VM/container placement and migration
Networking virtualization in edge context
(1) A summary on SDN and NFV architecture
(2) NFV virtualization techniques
(3) NFV placement and migration in edge computing
Future Research Direction

Fig. 1. The organization of the paper.

and networking resources at the edge network and mapping up-to-date studies to provide taxonomies in terms of virtualization type, cost function, objective function, solutions, and so on. (4) Identifying the potential research direction for virtualization of computing and networking resources in IoT frameworks that include *application*, *gateway* and *hardware* layers as discussed later. The indicated subjects shape the structure of the paper as shown in Fig. 1, in which the topics in the middle boxes in respect to computing and networking virtualization can be read independently.

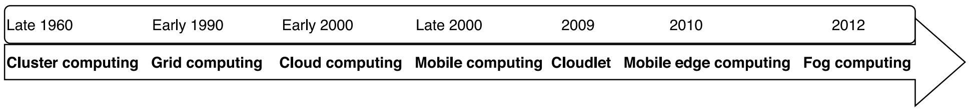


Fig. 2. The evolution of computing utilities.

The remainder of this paper is organized as follows. Section 2 introduces different cloud and edge computing paradigms, followed by the features and pillars analysis in Section 3 to understand the reasons behind moving from cloud to edge computing paradigms. Section 4 reviews and compares different virtualization techniques and identifies the selection factors of these techniques for IoT frameworks. Placement and migration of virtualized computing resources at the edge computing are discussed in Section 5. Section 6 discusses deployment and management of virtualized networking resources in the context of edge computing. Section 7 outlines the potential directions for future research followed by a conclusion in Section 8.

2. Cloud and edge computing paradigms

The study of distributed computing can be traced back to the early 1960s, when cluster computing emerged in the client server era, as shown in Fig. 2. The second milestone in the context of computing paradigms was in the early 2000s known as the cloud computing era. Then, the concept of cloud computing has been extended to the edge computing as refereed to the edge computing era. In the following, we discuss the last two computing eras and the reasons behind the transition between these two computing paradigms.

2.1. Cloud computing paradigms

Cloud Computing (CC) provides virtualized infrastructures, platforms, and software services over the Internet [42]. Infrastructure as a Service (IaaS) allows users to access computing, storage as a standalone VM. Platform as Service (PaaS) supports customers to develop, manage and run applications without dealing with software configuration and maintenance related issues to those applications. Software as service (SaaS) hosts applications and facilitates them with the needed infrastructures. All these cloud service models are identical in *elasticity*, *on-demand provisioning* and *pay-as-you-go model* [75]. Cloud computing supports *flexibility* in infrastructure scalability, resource diversity, level of access control and security features. It also provides *efficiency* in data and applications access, data security and Quality of Service (QoS), eliminates *capital expenditure* and reduces *operation costs.*

However, being no exception, cloud computing imposes *high latency* and *jitter* on applications due to sending and receiving data across wide area network (WAN) [94]. Second, it makes *vendor lock-in* as a major impediment to users who are vulnerable to any surge in services price, and changes to regulations and policies imposed by cloud providers [18]. Last but not the least, cloud computing supports *limited control* for users to manage and monitor their services. To overcome such limitations, users resort to *Spanning Cloud Computing (SCC).*

Spanning Cloud Computing (SCC) supports multiple datacenters deployment and extricates users from drawbacks of relying on a single datacenter, and explicitly or implicitly puts together the merits of different cloud providers [229,235]. SCC enables users to avert *vendor lock-in* which may lead to users' business becomes more resilient to distributed denial of service (DDoS) attacks. This is because if one datacenter goes down, users can utilize others. SCC also allows users to have a larger set of datacenters selection, which leads to latency and jitter reduction [249]. Furthermore, SCC offers users to have competitive options in terms of performance [193,235] and monetary cost [142,142].

Despite these benefits, we believe that SCC is a double-edged sword and suffers from the complexity of managing and monitoring services which require specific tools and software that may result in more challenges in cost management. It also potentially is a threat from security perspective due to revealing more access points to hackers. In summary, SCC enhances some performance metrics (e.g., reduction in latency, jitter and monetary cost) but it has still a big gap to meet *stringent latency*, *low jitter* and *mobility services* for IoT applications.

Cloud-based Content Delivery Network (CCDN) brings more and closer cloud-based servers to users to enhance *redundancy*, *performance* and *flexibility* [201,242]. CDNs reduce response time and jitter through these edge servers in Point-of-Presence (PoP) locations and reduce traffic on a core network via finding the best route to send requests to edge servers. Hence, they improve response time further in comparison to SCC but they still cannot meet the requirements of IoT applications.

Therefore, despite improvements in some features through the transition from cloud computing to SCC and then to CCDNs, all cloud-based paradigms are generally deficient in the required capabilities to serve IoT applications' overwhelming demand for handling and processing a tremendous amount of data in near real time. To fulfill the requirements of IoT applications, there is a need of edge computing ecosystem to support the processing of big data in the vicinity of the data sources.

2.2. Edge computing paradigms

In contrast to cloud computing, edge computing offers constrained and decentralized infrastructure resources, while bringing the required resources closer to data sources/edge and averting the requirements of sending data to a centralized cloud. Thus, edge computing reduces latency, jitter and load on a network core and improves security through storing data in on-premises infrastructure. The definition and location of *edge* are a controversial issue among researchers [261]. Some of those believe that *edge* is IoT-connected devices with a limited resource capabilities that process the collected data. The other researchers consider edge as a concept that moves processing to data sources. We believe that the *edge of a network and its location highly depend on the context of applications deployment*. An *edge is a logical border and it can be changed as the data consumer and data generator/provider are varied.* For instance, in the context of connected vehicles, a car is an edge where the location of data collection and process are the same, while in the context of MEC, radio access network (RAN) is an edge computing that processes a user's data. At a larger scale, a server in CCDN is an edge that process a user's requests. In the following, we discuss different paradigms of edge computing.

Mobile Cloud Computing(MCC) becomes popular as the proliferation of mobile devices necessitate to efficiently manage constrained resources. The nature of such resources attributes to the synergy between cloud computing and mobile devices as Mobile Cloud Computing (MCC) [72]. MCC overcomes several limitations from users' perspective like battery life, computation

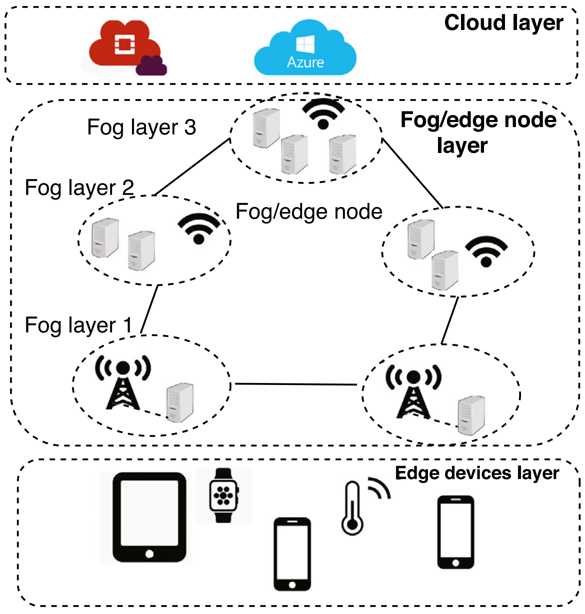


Fig. 3. Fog Architecture.

power, memory limitation especially for running data-intensive mobile-based applications [261]. Nevertheless, the deployment of cellular communication to transfer data to/from the network core on a long distance imposes high latency and jitter, as well as overhead on core network. Instead, the viable solution is to compute, analyze, and filter data close to data source, which paves the way towards Fog computing (FC) [261] and Mobile Edge computing (MEC) [133].

Fog Computing (FC) [40] pushes the resources down to IoT devices by enabling data processing, analyzing, managing and filtering at the edge of a network. FC allows computing and storage resources to scatter along the path between IoT devices and cloud computing depending on the required QoS. Thus, the notion of FC is broad and includes any devices with or without computing, storage and network capabilities. These devices vary from single board computers (SBCs), wireless/wired access points, network devices (switches, gateways), base stations, LAN networks, to any resource rich nodes. Fig. 3 shows hierarchical architecture of FC, where cloud is in the top, devices with very low richness are at the bottom, and fog nodes or Mobile Edge Computing (MEC) servers are in the middle. The trend of data management in fog ecosystem is of data collection via sensors and sending it to SCCs to act accordingly by actuators. For instance, in an agricultural automated irrigation system, sensors installed in agricultural farm collect data in terms of temperature, soil humidity and wind speed and send them to SCBs for analysis and taking proper actions, e.g., irrigating for 20 minutes. This trend makes FC suitable for IoT applications where mobility, low latency, location awareness requirements cannot be supported by cloud computing paradigms. This trend of data processing also makes FC not to have continuous connection of Internet since the updated data can be sent to cloud later. However, the data with high volume requiring heavy processing should be sent to a cloud to make long term and established decisions based on the available data.

Mobile/Multi-access Edge Computing[1] is an extension of MCC and enables cloud-based resources and services in the proximity of users. MEC is a subset of FC and provides virtualized

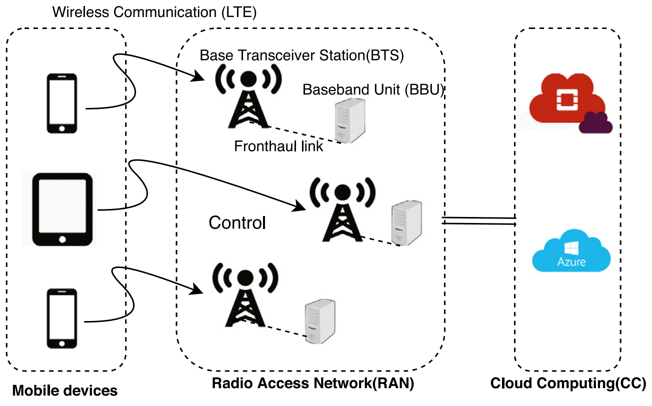

Wireless Communication (LTE)
Base Transceiver Station(BTS)
Baseband Unit (BBU)
Fronthaul link
Control
Azure
Mobile devices
Radio Access Network(RAN)
Cloud Computing(CC)

Fig. 4. Simple Architecture of Mobile Edge Computing (MEC).

computing resources with moderate capabilities in close proximity of the resource constrained mobile devices. These resources are termed MEC for which servers are co-located with base station (Fig. 4) and deployed at a stationary location such as a train station, stadium, shopping center or at a mobile location like car. The synergy between 5G and MEC supports significantly *faster data* (up to 10 Gbps), *ultra-low latency*, and more *connected world*. The evolution of Radio Access Network (RAN)[2] architecture leads to different MEC paradigms. For 1G and 2G cellular networks, RAN had all-in-one box including analogue, digital and power functions located in a dedicated room with all facilities such as air conditioning and power supply. From 3G onwards, the remote radio head/unit (RRH)[3] was separated from BBUs[4] through fiber cable or microwave and it is called D-RAN [93]. D-RAN includes a Base Transceiver Station (BTS) with a local dedicated BBU, which is associated with one cell. In D-RAN, *handover* is time consuming due to long distance between BBUs, and management cost is high as the number of users increases [13,49,54]. Thus, D-RAN cannot meet the requirements of IoT applications.

To address such issues, C-RAN [49] offers virtualized BBUs to decrease OPEX and CAPEX costs [238], and centralized BBUs to improve network performance and bandwidth rate. However, C-RAN shifting computing processing into a centralized BBU demands an interconnection fronthaul links with high bandwidth, which confines it to serve IOT applications. To cope with such drawback, a heterogeneous C-RAN (H-CRAN) [126,183] decouples data and control planes through High Power Nodes (HPNs)[5] to reduce fronthaul load and increase data transmission rate. Like C-RAN, H-CRAN cannot meet the requirements of IoT applications because it still exploits centralized resources that impose a heavy load on fronthaul links with the increment of users.

To further mitigate load on fronthaul links, processing and signaling resources are moved adjacent to users through UEs and network devices such as RRHs. This design leads to *distributed* and *centralized* Fog/F-RAN [28]. In the distributed F-RAN, some computation, storage and resource management functions are drifted from BBU to UEs, while in the centralized one resources are controlled and managed with exploitation of SDN and NFV. Thus,

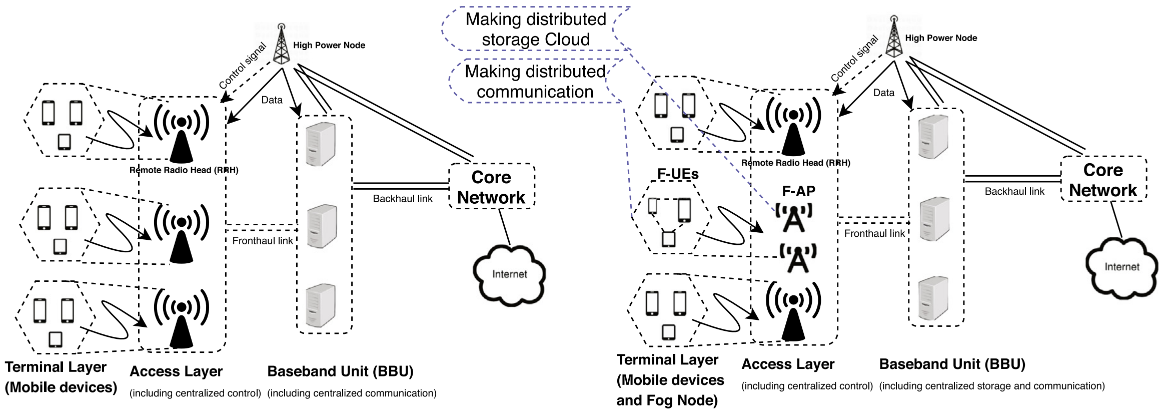
[1] The abbreviation of MEC is also used for multi-access edge computing which is beyond mobile devices and is applicable to use cases such as Augmented Reality (AR) and Virtual Reality (VR).

[2] RAN, as a part of cellular network communication system, makes a connection between mobile devices and backhaul network through the generations of mobile communications (1G-5G).

[3] RRH is a remote radio transceiver that connects to an operator radio control panel through electrical or wireless interface. It is used to extend the coverage of a BTS/NodeB/eNodeB.

[4] BBU dynamically allocates network resources to its associated RRH, and is connected to RAN through *fronthaul* links.

[5] HPN is utilized to support seamless coverage and execute control plane functions.



High Power Node
Control signal
Data
Remote Radio Head (RRH)
Backhaul link
Core Network
Fronthaul link
Internet
Terminal Layer (Mobile devices)
Access Layer (including centralized control)
Baseband Unit (BBU) (including centralized communication)
Making distributed storage Cloud
Making distributed communication
F-UEs
F-AP
Terminal Layer (Mobile devices and Fog Node)
Baseband Unit (BBU) (including centralized storage and communication)

Fig. 5. Architecture of H-CRAN (left) and F-RAN(right), F-UEs: Fog User Equipments, F-AP: Fog Access Point.

Table 2
Different Paradigms of Mobile Edge Computing (MEC).

Feature	D-RAN	C-RAN	H-CRAN	F-RAN
Architecture	NA	Centralized	Centralized	Centralized, Decentralized
Latency	Very high	Very high	High	Very low
Fronthaul load	Low	Very high	High	Very low
CAPEX/OPEX costs	Extreme high	High	Very high	Medium
Resource deployment	BBU, RRH	BBU,RRH	BBU, RRH HPN	BBU, RRH, Fog-node
Data and Control plane separation	No	No	Yes	Yes
Computing virtualization	No	Yes	Yes	Yes/No
Network virtualization	No	No	Yes	Yes/No
Resource proximity	Far	Close	Close	Very close
Cellular network utilization	3G-4G	3G-4G	4G	5G

F-RAN reduces latency further and makes resources management easier compared to D-/C-RAN and H-CRAN.

As discussed above, the aim of the changes in the architecture of edge computing paradigm is to reduce latency, jitter and load on network fronthaul and backhaul. This makes edge computing paradigms more suitable for IoT applications. As depicted in Figs. 4 and 5, apart from D-RAN, the architecture of all MEC paradigms consists of three layers. *Terminal layer* consists of mobile devices that host applications including *compute-intensive* (e.g., Augmented Reality), *latency-sensitive* (e.g., driverless cars, tactile internet applications), and *bandwidth-intensive* (e.g., mobile big data analytics). As can be seen, the main difference between MEC paradigms is in *access* and *network* layers. In C-RAN *control*, *caching* and *communication* are centralized in network layer (Fig. 4), while in H-CRAN control is centralized in access layer to reduce load on fronthaul link (Fig. 5, left-side). In F-RAN, storage and communication are centralized in network layer and distributed in access and terminal layers (Fig. 5, right-side).

With respect to the architecture of different MEC paradigms, we compare them from main features perspective as summarized in Table 2. F-RAN exposes resources at the edge of a network and it thus has the lowest latency, and H-CRAN follows the next. D-RAN and C-RAN impose the highest latency to users/devices, which depends on the distance between users and centralized resources/RAN in C-RAN/D-RAN. C-RAN increases load on fronthaul link more than H-CRAN, which in turn, raises it more than F-RAN in which the load reduces further through exploitation of edge network. Aside from D-RAN, all other MEC types leverage computing **virtualization** to enhance the efficiency of resource utilization. Due to the separation of data and control plane, H-CRAN and F-RAN can exploit **network virtualization**. Therefore, considering the features listed in the above table, F-RAN is the most suitable candidate to handle IoT applications.

Miscellaneous Edge Computing Approaches: The aim of all the discussed edge computing paradigms is to meet the requirements of IoT applications. In addition to the discussed edge paradigms, Cloudlet [204] is a type of edge computing that is located between edge devices and cloud computing in the proximity of users to address the drawbacks of MCC with exploitation of a trusted and resource-rich server in one hop away from users/edge devices (e.g., community places). Mobile ad-hoc Cloud Computing (MACC) is another type of edge computing and includes mobile edge devices which shares resources in a temporary and dynamic network via transport and routing protocols [101]. It is a decentralized infrastructure and suits to environments with lack of continuous internet connectivity [261]. Compared to FC and MEC, MACC consists of only mobile devices, making it more decentralized [261]. There are also some buzzwords such as *mist computing* [188] and *follow me cloud* [220] that more and less aim at closing resources to users. Liu et al. [128] presented open-source edge computing projects in academia and industry and draw a comparison between open-source tools for these projects.

3. Features analysis of cloud- and edge-based paradigms

In this section, we zoom out from details of each specific computing paradigm and analyze them with respect to their features and pillars as depicted in Fig. 6.

3.1. Feature-based comparison

To understand the objective of edge-cloud landscape as a whole, we investigate the following features.

(1) *Resource capacity and type:* It relates to amount and type of computation, communication and storage resources. As application providers move more towards edge-based paradigms, the capacity and type of resources reduce.[6] It is worth noting

[6] It should be noted that edge-based paradigms support more communication types such as Bluetooth, WiFi, cellular, ZigBee, while cloud-based paradigms mostly exploit wired connection.

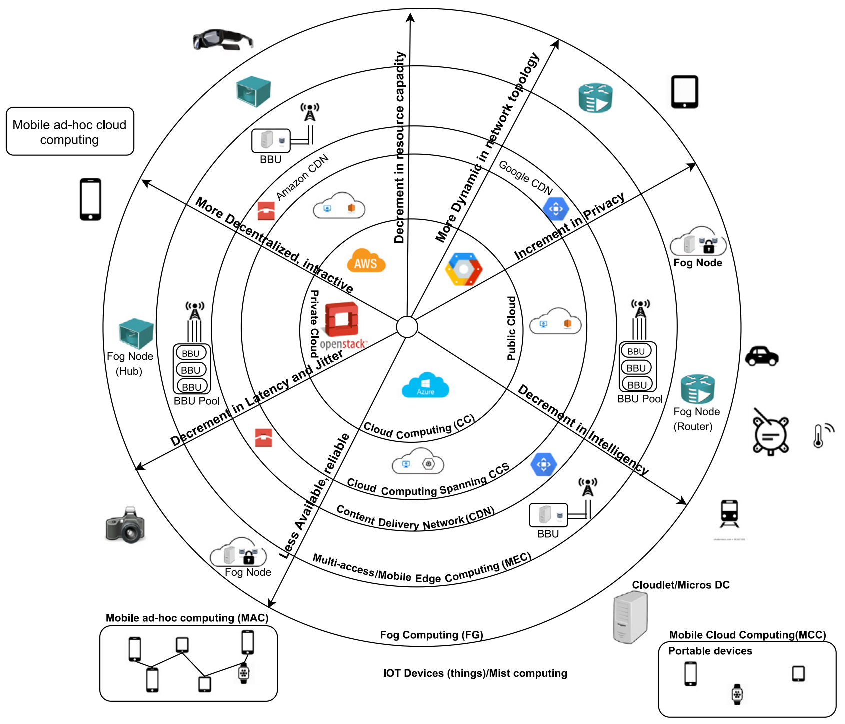

Decrement in resource capacity
More Dynamic in network topology
Increment in Privacy
More Decentralized, intractive
Decrement in Intelligency
Decrement in Latency and Jitter
Less Available, reliable
Private Cloud
Public Cloud
Cloud Computing (CC)
Cloud Computing Spanning CCS
Content Delivery Network (CDN)
Multi-access/Mobile Edge Computing (MEC)
Fog Computing (FG)
IOT Devices (things)/Mist computing
Amazon CDN
Google CDN
Mobile ad-hoc cloud computing
Mobile ad-hoc computing (MAC)
Mobile Cloud Computing(MCC)
Portable devices
Cloudlet/Micros DC
Fog Node
Fog Node (Hub)
Fog Node (Router)
BBU
BBU Pool

Fig. 6. Landscape of Cloud and Edge Computing Paradigms.

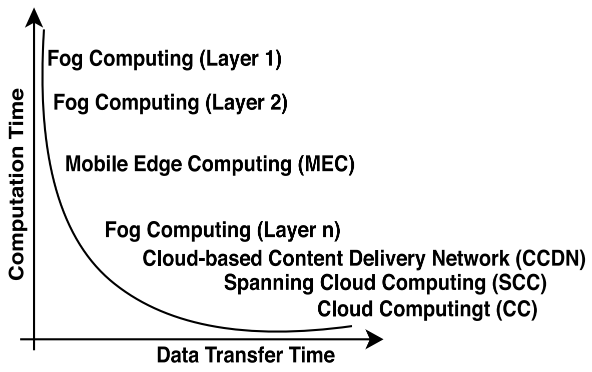


Fig. 7. Computation time vs. data transfer time in different computing paradigms.

that SCC is an exception that offers different types of resources from different cloud providers. The limitation of resources means application providers need to make a compromise between computation and data transfer time as illustrated in Fig. 7. Thus, real-time and short-term decisions are taken at the edge of a network; otherwise data is sent to cloud-based infrastructure.

(2) *Decentralized topology:* It refers to the arrangement of nodes (*every of things*) and connection in networks. More movement towards edge computing, more decentralized and distributed in network topology. This feature allows devices/resources to dynamically join and leave network in edge computing compared to those in cloud-based paradigms. In fact, from users perspective, network topology of cloud-based paradigms is a black-box, although a logical network topology in SCC can be defined such as a tree topology in which master and slave nodes respectively are at the root and non-root level of a tree. In edge-based paradigms, network topology can be a complex graph, where MACC is the most dynamic in network topology due to including only mobile devices with the highest degree of decentralization [261]. The network topology has significant effects on performance metrics such as execution delay required to offload an application to edge computing resources.

(3) *Latency and jitter:* Latency is the most important reason behind the deployment of an application at network edge [40,47]. As applications are more shifted from cloud-based to edge-based paradigms, they incur less *latency* and *jitter*. Cloud-based paradigms impose several hundreds of milliseconds latency which depend on the distance between client and cloud, cloud datacenter provider, route of the network traffic, among others [155]. Deployment of SCC can reduce latency 80% and even further through CCDNs [249]. Even several tens of milliseconds latency are still high for real-time applications, which tolerate at most hundreds of nanoseconds or less than ten milliseconds. High jitter has a negative influence on performance of real-time applications especially beyond 100ms [250]. Edge-based paradigms reduce jitter since they avoid sending data over WAN network.

(4) *Intelligence:* Full life-cycle data service consists of the following sub-services [273]. *Data pre-processing* includes aggregation, filtering, and cleaning raw data. *Data analysis* provides decision making based on optimization approaches and statistical models. *Data distribution and execution policy* executes predefined rules based on analyzed data locally in network edge or sent it to a cloud. The first sub-service is usually managed in edge devices, while the two later sub-services are conducted either in a network edge or a cloud. The selection of each paradigm depends on the required *response time*, *resources capability*, and the *size of data* leveraged. Thus, as the capability of the resources increase the intelligence strengthen raises from edge- to cloud-based paradigms as shown in Fig. 6

(5) *Availability and reliability:* As long as we move towards edge-based paradigms, availability and reliability decrease as depicted in Fig. 6. This is because components of a system cooperatively provide a service. It should be noted that SCC is an exception in this respect because replication of computing and data across multiple-datacenters increase availability [140]. MACC has the lowest availability among edge computing due to freely attached and detached mobile nodes to ad hoc networks.

(6) *Security and privacy:* Moving from cloud- to edge-based paradigms increases privacy and reduces security. This is because (i) edge-based paradigms eliminate data access through third party especially in FC due to storing data under control of users, and (ii) cloud-based paradigms are more strengthen in physical cyber-security and deployment of centralized and heavy security mechanisms. Due to the nature of edge computing, it is required to design flexible, lightweight security mechanisms so that attacks and failure to a certain extent should be handled and recovered in the least amount of time. Classical trust-based security mechanisms such as *policy- and SLA verification-based* [112] are not an effective way to utilize a large number of heterogeneous edge devices. Thus, deployment of security mechanism with less authentication principles (e.g., white-listing functions [78]) can be a viable solution.

(7) *Resource utilization:* To understand the usability of edge computing for a specific application, we need to define performance metrics in terms of resource utilization (processing, storage, bandwidth, I/O, energy) and throughput [20]. As we move more towards the edge computing, these performance metrics are more critical from users perspective due to tight constraint on processing power, storage and RAM capacity, as well as battery lifetime. Among these, battery life is the most important concern from user perspective in the context of edge computing compared to the cloud computing that exploits both brown and green energy [146,255]. To relieve this concern, the solutions can be varied from computing offloading strategies, lightweight design OS for mobile devices, hardware modules (e.g., CPU and GPU) to spread workloads across edge-cloud computing [53,232]. Nevertheless, cloud and edge computing are common in many of these performance metrics that are evaluated mostly via simulators. Since these simulators mostly consider many assumptions (which are not compatible with real environment), it would be better to evaluate these parameters in a real implementation for edge computing. This evaluation is more in accordance with the real-time data generation in edge computing required for online action to a particulate event. Interested readers are referred to more details in respect to the evaluation and definition of performance metrics for cloud and edge computing in [21].

3.2. Pillars-based comparison

We define pillars[7] as the essence of edge-based paradigms in terms of constraints and objectives. The pillars are as follow.

(1) *Constrained-resource devices:* These devices include a range of sensors, actuators, gateways, routers and servers with medium capabilities (e.g., MEC server). They process data closer to data sources, reduce data transfer time and provide intelligence to react to the environmental events. However, such devices restrain to deploy heavy virtualization techniques and data processing. Thus, the former lack mandates to use lighter virtualization techniques, and the latter one necessitates a synergy between edge and cloud paradigms.

(2) *Ultra-low end-to-end (E2E) latency*: This is one of the essential pillars that inspires users to make the resources close to data sources. This requirement is a necessity for services such as car-to-car communication, remote operations and factory automation. The following ways can reduce E2E latency. (i) *Enhancement in cellular communication*: 5G makes major advance on bandwidth capacity (100 Mbps–10 Gbps), improved connectivity (to more than 100 billion devices), reduction in energy consumption (1000 times lower) and mobility support (for users/devices with speed of at most 500 km/h).[8,9] (ii) *Move resources closer to users*: As distance between edge devices to edge computing is shorter, E2E latency is less. It is worth noting that E2E latency depends on the required time for sending data to/from edge devices and processing task/data in edge server [133]. Thus, FC and MEC are the most fitting to reduce E2E latency. (iii) *Offloading tasks to edge computing*: As we will discuss later, this is one of the most vital strategy to reduce E2E latency by optimizing the deployment of under-utilized servers, and less congested links between edge device and edge server.

(3) *Mobility Services:* This is another pillar that motivates users to leverage edge computing paradigms. Mobility can be one *direction* or *mutual*, where in the former one device is mobile and other is fixed as in MEC [72,133] and Cloudlet [204], while in the latter both devices are mobile as observed in the context of FC such as Vehicle to Vehicle (V2V) communication [264]. Mobility can be considered for computation, data and users. In one-direction mobility, most data and computation are transferred from a less capable device to a more capable one to speed up computation and meet QoS. Whilst in mutual mobility, users/devices make close to each other or relay on MEC to speed up procuring data required. Moreover, in mutual mobility resource discovery is more challenging since each side is mobile and required specific QoS. Mobility services must provide stable connectivity between mobile devices through *handover/handoff* [91,92] that is initiated via network or users based on factors such as application type, required bandwidth, delay preferences, power consumption, network load and estimated data rates [72,133]. The handover operation is abstracted in the form of a VM, which is placed close to mobile devices. Obviously, this operation is more complex for mutual mobility since VM placement is affected by two mutual mobile devices as required for V2V wireless communication [17].

(4) *Virtualization:* Virtualization[10] is the essence of edge-cloud environments and abstracts computing and network resources [225]. Virtualization plays a key role in CAPEX and OPEX reduction, resources allocation to the new services and reclaiming them as no longer needed. In contrast to the cloud data centers, the characteristics of edge computing impose several challenges to virtualized computing and network resources. (i) *Resource-constrained* edge devices/fog nodes mandate to deploy lightweight virtualization techniques in terms of memory footprint. (ii) Mobility of edge devices/users implies to deploy agile virtualization techniques in terms of memory footprint and instantiation time to easily migrate VMs from one node to another. (iii) The nature of edge devices distribution and wireless and wired communication make a requirement of global management and control of resources.

We notice that virtualization techniques have been well studied and developed for the well-rich resources (e.g., cloud data centers). Nonetheless, as summarized in Table 4, most edge paradigms support computing and networking virtualization, and these virtualization techniques should be tailored for them to make compatible with their indicated limitations. Due to high mobility of devices, VM migration in edge computing should be revised to make it agile for time critical applications. Networking virtualization providing via SDN and NFV framework [39] needs to be efficient in terms of bandwidth, energy, and performance,

[7] In this paper, *pillars* of the edge computing refer to the most imperative objective functions (i.e., resources capacity, ultra-low latency, and mobility services) and requirements (i.e., virtualization support).

[8] Although 5G is in initial state of deployment, Samsung Electronics expect to leverage 6G as early as 2028. It is expected 6G provides two times more energy efficient, 100 times the peak data rate of 5G, and one-tenth the latency of 5G.

[9] The Next Hyper Connected Experience forAll:https://cdn.codeground.org/nsr/downloads/researchareas/6G%20Vision.pdf

[10] In addition to *virtualization*, three other challenges that should be addressed in an edge computing platform and fall out of this paper scope are *programmability*, *interoperability* and *elasticity* [159].

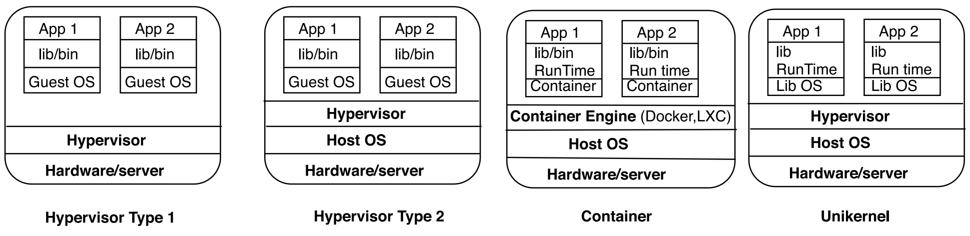


Fig. 8. A comparison of computing virtualization techniques from architecture perspective in both cloud and edge computing.

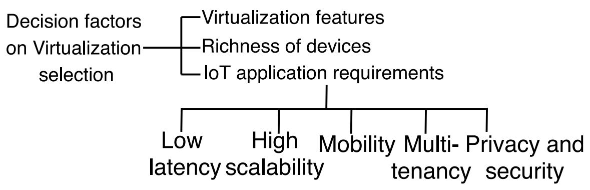


Fig. 9. Decision factors for the selection of virtualization technique in edge computing.

and also compatible with legacy devices. Such virtualization hides the complexity of IoT platform and makes efficient IoT big data analytic in the collaborative cloud-edge paradigms. The rest of this work discusses virtualization of computing and networking resources at the edge network for IoT applications (see Table 3).

4. Virtualization of computing resources

The concept of virtualization came into play in mid 1960s [85], when IBM required to run multiple tasks on a mainframe in parallel. Virtualisation allows to partition a single physical server into several virtual machines (VMs) in hardware level [215] (e.g., Xen [251] and KVM [121]). Hardware-level virtualization, dominated in cloud computing, cannot effectively be leveraged in edge paradigms. This is because these paradigms aim at delivering low latency, bandwidth efficient and resilient services to IoT applications through constrained-resource edge devices. Thus, it is required to scale down virtualization infrastructure in OS-level [215] (e.g., Docker[11] and LXC[12]) featured in low initialization time, small memory footprint, and image size. Merely usage of lightweight (i.e., OS-level) virtualization cannot meet the requirements of IoT applications as shown in Fig. 9. Thus, the decision on which virtualization technique to use depends on, as depicted in Fig. 9, *features of virtualization techniques*, *device capabilities* and *application requirements*. We discuss these factors in the following sub-sections.

4.1. Computing virtualization in edge computing

This section discusses virtualization techniques and provides a detailed comparison between them.

Hypervisor-based Virtualization provides hardware level isolation in two types [211] as depicted in Fig. 8. Type 1 hypervisor runs directly on the server/hardware without loading an underlying OS, while Type 2 hypervisor is installed on top of the underlying OS with the same purpose of Type 1. Type 1 is more efficient in security and latency due to the underlying OS elimination compared to Type 2 although it introduces more latency and vulnerability to security threats. In both types, each VM requires Bins/Libs,[13] and guest OS. Thus, hypervisor-based virtualization is subject to large image size and slow instantiation time.

Container-based Virtualization is an OS level isolation and overcomes constraints of the hypervisor-based virtualization through a container engine in which host OS kernel is shared among processes [24] (Fig. 8). Thus, container-based virtualization reduces the image size (several MB) and the instantiation time (several seconds) of a container, making it a lightweight virtualization that utilizes resources more efficiently than its counterpart. Containers offer near native computing and network I/O performance [71].

Sharing host OS confronts containers with two major issues. (i) *denial of service*: If one application consumes most of the OS resources, then other applications are deprived from the bare minimum resources required for continued operations. (ii) *exploiting kernel*: If an attacker is able to take control of a host OS, it can access all applications running on the host. Thus, containers compromise security for performance. Furthermore, a lack of full interoperability is another weakness of containers and impedes them to fully operate across different clouds. For instance, OpenShift [194], Red Hat's container-as-a-service platform, only works with the Kubernetes orchestrator [119] offered by Google.

Unikernel [135] is an executable lightweight virtualization image that runs on a hypervisor without requiring a separate OS as depicted in Fig. 8. It includes application code that executes a process and the required OS libraries for that application. Thus, unikernels are a single-purpose approach with smaller memory footprint and lower instantiation time. They also shrink the attack surface due to including used drivers, necessary I/O routines and the needed support libraries. Hence, they improve security features. With respect to performance, they run entirely in the privileged mode of CPU to avoid a context switch across the user-kernel boundary. However, some researchers criticize no clear definition behind unikernel memory footprint and library components included. Migrages[14] and ClickOS[15] are examples of unikernel.

A comparison of virtualization techniques: As shown in Fig. 8, the main difference between hypervisor- and container-based VMs is OS sharing. The same difference exists between containers and unikernels, but unikernels, in contrast to hypervisor-based VMs, include only libraries needed for an application. Thus, we make the following two remarks. (1) Unikernels are smaller in image size than containers, following by hypervisor-based VMs; As a consequence, unikernels and containers are more agile/portable to be moved while VMs are not. In practice, there is no significant difference in image size for VMs and containers that host data intensive applications. This is due to the size of OS kernel (about 10 MBs) is very less than the size of data (several GBs) hosted by VM and container. Therefore, the migration time

11 Docker container:https://www.docker.com/.
12 LXC container:https://linuxcontainers.org/.

13 Bin files directory include the executable programs that must be available for minimal system working, and *lib* files directory contain all helpful library files used by processes in the system.
14 Migrages:https://mirage.io/.
15 ClickOS:http://cnp.neclab.eu/clickos/.

Table 3
A comparison of features and pillars of cloud-based paradigms. Pillars are bold.

Feature	Cloud Computing(CC)	Spanning Cloud Computing(SCC)	Cloud-based Content Delivery Network(CCDN)
Proponent	Cloud providers	NA	Usenet
Architecture	Centralized	Graph-based	Graph, Hierarchical
Usage type	General	General	Content Delivery
Standardization	NIST, ITU	CSCC, ETSI	IEEE
Use-case	General	General	Content-delivery
Network connectivity	WAN	WAN	WAN, LAN
Resources type	DC	Mulitple-DCs	MD+DC
Resource capacity	High	Very High	High
Application(s) support	Computation-intensive	Computation-intensive	Data-intensive
Mobility support	No	No	No
Virtualization support	Yes	Yes	Yes
Proximity	Very Far	Far	Relatively far
Ultra-low latency	No	No	No
Location awareness	No	No	No
Security granularity	DC	DC	Participant data servers

NA: Not Applicable, DC:DataCenter, WAN: Wide Area Network, MD: Mobile Devices, ED: Edge Devices, ETSI:European Telecommunications Standards Institute, CSCC: Cloud Standards Customer Council, NIST: National Institutes of Standards and Technology, ITU: International Telecommunication Union.

Table 4
A comparison of features and pillars of edge-based paradigms. Pillars are bold.

Feature	Mobile Cloud Computing (MCC)		Mobile Edge Computing (MEC)	Fog Computing (FC)
	CloudLet	MAC		
Proponent	NIST	NA	Nokia, IBM	Cisco
Architecture	MD+CloudLet+DC	Distributed	Distributed	Decentralized/Hierarchical
Usage type	Mobile Devices	Mobile Devices	Mobile Devices	General
Standardization	NIST	NA	ETSI, 3GPP, ITU	OPenFog Consortium, IEEE
Usecase	Mobile applications	Disaster relief	mission-critical	IoT applications
Network connectivity	WAN	Bluetooth, Wi-Fi, Cellular	WAN, cellular	All communication types
Resources type	MD+DC	MDs	RAN+MD+DC	Any devices
Resource capacity	Limited	Moderate	Moderate	Very limited
Application(s) support	Moderate Computation	Very low computation	Moderate computation	Low computation
Mobility support	Yes	Yes	Yes	Yes
Virtualization support	Yes	No	Yes	Yes
Proximity	Close	Very close	Close	Very close
Ultra-low latency	No	No	Yes	Yes
Location awareness	No	Yes	Yes	Yes
Security granularity	Mobile device, DC	Mobile device	DC, RAN	ED, network between ED and DC

3GPP: 3rd Generation Partnership Project.

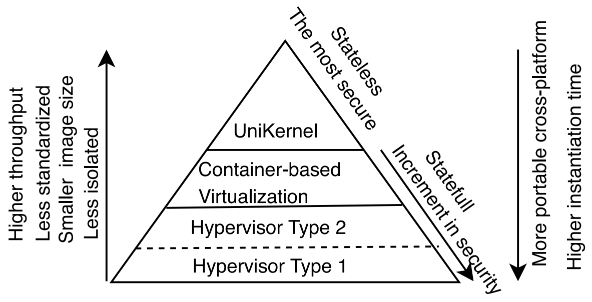


Fig. 10. Properties of different virtualization techniques.

of an individual VM or container is roughly same, but this time can be significant if a bulk of VMs or containers are migrated. (2) A container has the lowest isolation level due to exploiting sharing OS kernel, while other techniques provide better isolation level for VMs.

Fig. 8 also shows that unikernels are different in allocation of the required library components of OS to each application. The minimization of library components number in unikernels might help them to be safer than hypervisor-based VMs. Some references believe that unikernels are a shortened version of hypervisor-based VMs [136,175]. In practice, with respect to the benefits of hypervisor- and container-based virtualization techniques, a better way is to have agility of containers and safety of VMs via hardware enforced isolation. In-Process Memory Isolation Extension (IMIX) [76] and Intel Software Guard Extensions (SGX) [102] are a set of security related code that is built into modern CPU to encrypt a portion of memory to protect VMs from accessing each other.

Fig. 10 compares virtualization techniques based on the following properties. A smaller *image size* requires less resources to be hosted and less time to migrate. The image size of a VMs, containers and unikernels typically reaches to hundreds MBs, tens MBs and several MBs respectively [159]. A shorter *instantiation time* brings less latency and it thus improves Quality of Experience (QoE) from user's perspective. This value for VMs, containers and unikernels is several seconds or even minutes, hundreds milliseconds and tens milliseconds respectively [159]. A low powered hardware resources require a lighter virtualization technique for hosting a VM and less bandwidth for VM migration if needed. Safety relates to the isolation level of the address spaces used by VMs running on the same hardware. A VM technique can perfectly achieve this goal if it does not allow penetration of security threats through co-located VM [266]. The capability of *storing persistent data* is another criterion that makes VM more suitable in comparison to lightweight virtualization techniques.

Table 5
Edge devices with the capability of virtualization techniques.

Edge devices type	Virtualization technique	Examples of devices
Dumb devices	NA	Sensors, actuator
Low-capability devices	Lightweight	Raspberry Pi, SBCs, Camera [107,157]
Medium-capability devices	Most lightweight	Cloudlets [204]
High-capability devices	Most heavyweight	MEC servers [72,133]

4.2. Richness of resources

As shown in Fig. 9, the second factor that impacts the selection of virtualization technique is *richness of resources*. In contrast to the cloud computing, edge computing exploits constrained-resource devices with low richness of CPU, RAM and storage. With respect to the capability of resources, as summarized in Table 5, we classify the edge devices as below.

(1) *Dumb devices* lack computing resources to be virtualized and it is thus exploited in the form of bare metal. For example, sensors and actuators collect data from environments based on their functionality.

(2) *Smart devices with low-capability resources.* They are labeled as single-board computers (SBCs)[16]– such as Raspberry Pi (RPi)– with scarce resources in terms of CPU, memory, storage and network [192]. It is efficient to deploy lightweight virtualization (either container/unikernel or both) for this type of device, which depends on the requirements of IoT applications. Unikernels are an elite choice to deploy in these devices to process and analyze data close to data sources (e.g., augmented smart house or vehicles) by using Machine Learning (ML) models. Container-based virtualization can be deployed in such devices with almost negligible effects on performance compared to native performance running on bare metal without any virtualization [157]. A combination of both virtualization techniques can be utilized in a cluster of SBCs [106] to achieve multi-tenant (provided by containers) and high safety (provided by unikernels located in containers).

(3) *Smart device with medium capability resources.* This type of devices is expected to support a limited number of applications requiring moderate resources. These devices are deployed either in closed (e.g., shopping malls) or open (e.g., stadium) environments to serve either stationary or mobile devices. It may be efficient to deploy hypervisor based virtualization for serving stationary users but it should be tailored for mobile users running time critical applications that require a smooth handover defined as switching radio communication from the serving cell such as small cell base stations (SCeNBs) to another one. Since communication between two devices is not available during handover, hypervisor based virtualization should be improved in initial VM provisioning and migration. VM *handoff* [91,92] might be a viable solution where a base image is stored in the device and only the overlay of VM (the difference between the VM in the source and destination devices) is migrated. Cloudlets [204] and MEC servers can use such virtualization technique.

(4) *Smart device with high-capability resources.* These devices are resource rich hardware and handle delay tolerant applications on the order of a few seconds or longer. A set of such servers are utilized in Cloud-RAN (C-RAN) [49] that takes the advantage of hypervisor-based VMs in which Docker-based containers run. The combination of both techniques enables C-RAN (i) to host heavy applications or a bunch of them relating to the same third-party demanding a high degree of safety via VMs, and (ii) to offer mechanisms for packaging and agile movement in MEC collaboration space.

[16] Commodity SBC is a complete computer built on a single board and has sufficient power to run OS and workload. A cluster of SCBs replicates some features of large datacenter cluster with low cost in order to host a range of IoT applications [105].

4.3. Iot application requirements

As depicted in Fig. 9, the third factor that affects the selection of virtualization technique is the *requirements of IoT applications*. There are numerous IoT applications that provide three main services by deploying in the edge computing [38]. *Enhanced Mobile Broadband* (eMBB) service is suitable for applications that require high data rates across a wide range of coverage area. *Ultra Reliable Low Latency Communications* (URLLC) service is appropriate for applications that need mission critical communications. *Massive Machine Type Communications* (mMTC) service suits for applications that connect to a large number of devices. The power of edge paradigms supports these services, where each service includes different IoT applications with various requirements.

Fig. 11 shows that *latency* and *scalability* respectively are top priorities for URLLC and mMTC services, while both requirements are the top priority for eMBB services. The importance of remaining requirements depends on the type of the application. *Mobility* feature is the common requirement for all IoT services and has essential effect to consider deployment of lightweight virtualization. *Privacy and security* can be important for applications that process sensitive and personal information (e-health application), and show track of users (e.g., analyzing energy consumption of users via smart grid application). This requirement inclines the use of heavyweight virtualization. *Multi-tenancy* requirement is critical for processing sensitive and personal data (e.g., e-health and smart city applications) and non-critical single controlled data (e.g., autonomous vehicles or smart grid). Hence, multi-tenancy implies the use of virtualization technique with higher isolation.

Solely deployment of IoT applications at the edge network cannot meet the stringent requirements, discussed above, for the critical IoT services. To achieve such requirements, it is vital to optimize resource management to reduce response time, and energy consumption, and to improve resource utilization. We believe that resource management in edge environment is more complicated compared to the one in cloud computing due to distributed resources and hierarchical levels in richness of resources. Recently, for example, some studies exploited AI and blockchain to optimize performance metrics and energy consumption in the edge-based healthcare applications [84,217,231,232]. For more details about resources management to meet QoS for particular application, readers are referred to [81,100].

Therefore, based on the *prioritized requirements*, *richness of resources* and *features of different virtualization types*, we can select a virtualization technique to abstract resources. Relying on a single virtualization technique cannot meet all the requirements of an application and a combination of virtualization techniques can be envisioned [106,158]. For example, a combination of hypervisor based VM and unikernels is suitable for applications that need multi-tenancy. A combination of container based virtualization and unikernels is appropriate for applications with a need of high mobility.

4.4. Virtualization of IoT platforms

The aim of edge cloud computing is to facilitate IoT applications to meet their needs through efficient utilization of edge

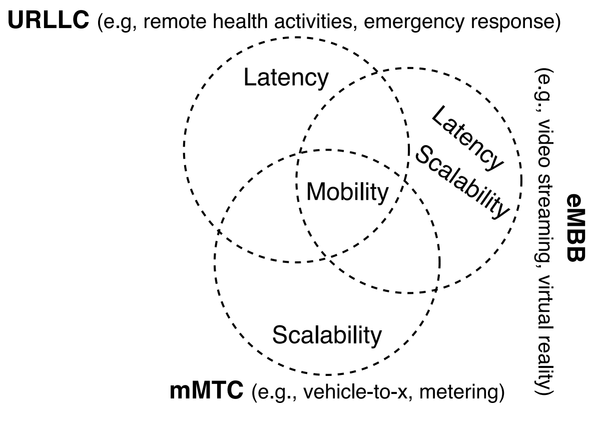

URLLC (e.g, remote health activities, emergency response)
Latency
Mobility
Latency
Scalability
Scalability
eMBB
(e.g., video streaming, virtual reality)
mMTC (e.g., vehicle-to-x, metering)

Fig. 11. The top requirements of IoT services.

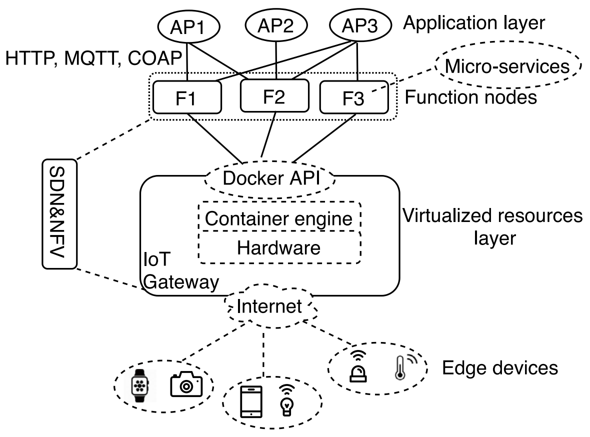


Fig. 12. A simplified schema of IoT framework.

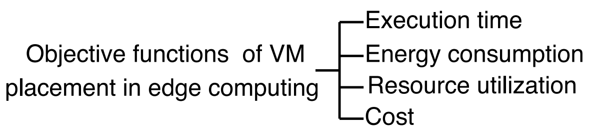


Fig. 13. Objective functions of VM placement in edge computing .

devices, computing and network resources. To this end, virtualization allows to share devices, resources and *micro services/functions* between IoT applications. Micro services provide transformation from monolithic cloud computing to the set of functions [95], which, in turn, to create large, complex and horizontally scalable IoT applications. As discussed above, lightweight virtualization, specially containers, is the most promising candidate for hosting micro services within IoT devices and share them between IoT applications in a particular domain such as smart houses.

Fig. 12 illustrates a simplified schema of IoT framework virtualization, which consists of three layers. Applications layer represents micro services built for a particular IoT application or shared between IoT applications. As an example, applications AP1 and AP3 share micro-service F1. Virtualized resources layer represents IoT Gateway that connects IoT devices with the application layer via, for example, Docker API. The hardware layer that includes IoT devices and fog nodes with richer resources. IoT gateways can be envisioned including application services (orchestrator (e.g., Kubernetes,[17] Docker Swarm,[18] and Mesos[19]), database, and web server), and underlying software components that virtualize IoT devices. IoT gateways, as summarized in Table 6, mostly leverage Docker container to instantiate fast process with small footprint and to provide high density multi-tenancy.

IOT gateway design should expose several features to effectively manage dynamic IoT environment [161]. *Scalability* represents dynamic joining and leaving of IoT devices simultaneously being connected to an IoT gateway. *Interoperability* of an IoT gateway makes possible data transformation via different protocols such as HTTP, MQTT and COAP [61] to support interactions between different applications and systems. *Isolation and multi-tenancy* of the IoT gateway provided by lightweight virtualization techniques facilitate IoT applications to share micro services and improve resources utilization [161,172]. However, some of the reviewed studies did not meet this feature for the designed IoT gateway [58,60,89]. Table 6 shows virtualization techniques exploited in the IoT frameworks.

5. Performance oriented virtualization optimization techniques in edge computing

5.1. VM placement

From a user's perspective, one of the imperative issues relating to a fog node is to select VM(s) to offload applications to speed up the process of computation and save energy consumption at edge devices. The fundamental questions with respect to the computation offloading are: *when*, *where*, and *which parts of* an application should be deployed at VM(s) in fog nodes?

Basically, two ways can be envisioned to offload an application: *full* and *partial* [133]. Full (resp. partial) offloading refers to offloading *all* (resp. *some*) parts of an application that depends on two primary factors. The first factor is whether an application can be divided into several parts for migration from edge nodes/devices to fog nodes. If so, which parts have *one- or double-sided dependencies* and which parts have not [143]. Independent parts can be offloaded in the same or different fog nodes and simultaneously processed in parallel. Otherwise, all dependent parts are offloaded to the same fog nodes. The second factor is the *amount of data* to be transferred into a fog node, which is known beforehand (e.g., image processing and sound recognition) or is unknown (e.g., augmented reality(AR)).

A decision about *when* an application should be partially or fully offloaded depends on *mono* or *multi-* objective function, where the former optimizes applications offloading based on a single objective function and the later one based on several objective functions [1]. As shown in Fig. 13, the imperative objective functions are *execution time*, *energy consumption*, *resource utilization*, and *cost*. *Heterogeneity*, *mobility*, and *geographical location* of edge resources make differentiate and complicate the optimization of these metrics in comparison with the ones in the context of cloud. As depicted in Fig. Fig. 14, *execution time* is the duration time for offloading data to a fog node, processing time at it, and duration time for sending the processed data back. Thus, the amount of data load on network links and the capability fog nodes are determining factors in whether an application should be offloaded. *Energy consumption* is the required energy to send data to a fog node and send back the results from it, which leads to save battery power on computation due to performing it on fog nodes. A trade-off between energy consumption on computation and data transfer determines whether to conduct computation offloading. Resource utilization relates to maximize the number of micro-services served over fog nodes [100]. Cost is defined for users and service provides, and it consists of communication cost

[17] Kubernetes: https://kubernetes.io/.
[18] Docker Swarm: https://www.docker.com/.
[19] Mesos: http://mesos.apache.org/.

Table 6
Virtualization techniques leveraged in IoT applications/frameworks.

Work	IoT application	Virt. techniques	Hardware resources	Objective
Morabito et al. [161]	IoT gateway	Container	SBC	An implementation of scalable and energy-efficient IoT Gateway with isolation and multi-tenancy features
Novo et al. [172]	Transport, agriculture	Container	RPi (model B)	Providing the capabilities of cellular networks to constrained networks while enabling the connectivity between wireless sensor networks and cellular networks
Li et al. [127]	General	Hardware-,OS-level	Sensor devices	Design a virtualization enabled fog computing for IoT
R. Morabito [157]	NA	Container	RPi	Performance and energy evaluation of lightweight virtualization instances on SBCs
Lee et al. [123]	NA	Container	RPi3	Network performance evaluation of container-based virtualization, which is lower than that of naive Linux
Morabito et al. [159]	Smart City, V2V	Container, Unikernel	RPi3	Applicability of lightweight virtualization techniques in smart city and vehicle communication platforms
Roca et al. [198]	General	NS	Sensor, router	Define Fog Function Virtualization(FFV) concept to provide flexibility and adaptability to run different applications on edge devices
Kakakhel et al. [107]	NS	Container	SBC	Evaluation of container-based live migration for stateful applications
Morabito et al. [160]	General	Container	RPi3	Performance evaluation of container-based virtualization for clustered IoT devices
Noronha et al. [171]	General	Container (LXC)	micro-processor	Performance evaluation of LXC on broad range of embedded micro-processors
Jang et al. [104]	Video analytics	Container	Camera	Video analytics at the edge computing to adapt cameras to environment changes
Ogawa et al. [174]	Smart city,	Container	Camera	IoT devices virtualization to share them between IoT applications to improve utilization of computing resources.
Harjula et al. [95]	General	Container	RPi3	A virtualized and decentralized nanoservice-based model
Celesti et al. [46]	General	Container (LCV)	RPi	Evaluation the overhead of container virtualization in an IoT device
Puliafito et al. [190]	General	Container	RPi3	Evaluation of container-based virtualization for cold and live migration techniques
Samaniego and Deters [202]	Smart-house	Unikernel	RPi2	Abstracting the complexity of IoT devices from user's perspective and creating many-to-many relations between users and devices
Saurez et al. [205]	Vehicular traffic	Container	Relion servers	Handling proactive migration of components of the application between fog nodes
Bellavista and Zanni [32]	Smart connected vehicles	Container	RPi1	Creating scalable and flexible fog nodes over constrained-resource devices
Chang et al. [47]	Video surveillance, 3D indoor localization	Container (LXC)	Smart phones, RPi	Bringing the capabilities of OpenStack to the edge devices
A. Krylovskiy [118]	General	Container	RPi2, RPi B+	Evaluation the overheads of containerized IoT gateways in terms of throughput and latency
Hong et al. [99]	Image processing	Container	RPi	Implementation of a fog computing platform and studying an efficient modules deployment on fog devices to maximize number of satisfied requests
Salikhov et al. [115]	Smart houses	Unikernel	Sensors network	Implementation of a prototype platform for running multiple concurrent applications over a network of sensors
Kaur et al. [110]	General	Container	NA	Architecture implementation for tasks selection and scheduling at edge network to optimize energy consumption leveraging a container migration technique
Karhula et al. [109]	General	Container	RPi3	Evaluation of a multi-container IoT gateway in terms of delay and the number of events in comparison to the non-containerized gateway
Gonalves et al. [86]	General	VM	Cloudlet	A simulation-based evaluation of proactive VM placement and migration based on mobility prediction
Dolui and Kiraly [63]	General	Container	ARM Cortex A53	Evaluation of a multi-container and micro service-based framework for IoT gateway using base image hierarchies and image layering to optimize in/cross-container performance optimization

(continued on next page)

Table 6 (*continued*).

Work	IoT application	Virt. techniques	Hardware resources	Objective
Alam et al. [11]	Smart home	Container	RPi3	Implementation of a modular and decentralized for IoT architecture using container and micro-services
Petrolo et al. [186]	General	Container	RPi2	Implementation of a semantic-based gateway for cloud of things, acting as an interface for users as well

Virt: Virtualization, NA: Not Applicable, NS: Not Specified, RPi: Raspberry Pi, V2V: vehicle to vehicle

to offload the components of applications and computation cost to process the task at fog nodes. The main difference between cost optimization in cloud and edge computing is that fog node might be possessed by different service providers. This complicates cost optimization in the edge context from user perspective although provides more available options to exploit cheaper services.

After taking decision on *when* and *which* part(s) of an application to be offloaded, VM allocation must be performed in a single or multiple fog nodes to process a task. For *non-partitioned* applications, VM allocation in a single fog node relies on the availability of VM resources and the required execution time for an application. If VM resources are not enough, a fog node should ideally delegate to the request to another fog node or to a centralized cloud based on the objective functions and constraints. For *partitioned* applications, VMs allocation, consequently, the optimization of objective function becomes more complicated as multiple fog nodes host workload [131,241]. Irrespective of partitioned or non-partitioned applications, this optimization problem in the edge context is NP-hard [200]. Approximation (as in [22, 221], heuristic [41]), and meta heuristic (i.e,Ant Colony [65], Genetic Algorithms [98], Particle Swarm Optimization [111]) are lightweight solutions to optimize objective functions in edge paradigms, which can be evaluated via simulator (CloudSim [43] and iFogSim [90]), analytical tools (Matlab, IBM CPLEX), and test bed environments (Grid'5000 [26] and Adhoc testbed [77]). Table 7 summarizes the objectives of IoT applications deployed in the virtualized edge computing. Note that the mobility in this table refers to the mobility of devices or users.

5.2. VM migration

VM migration refers to moving one VM from a physical node to another one to improve performance and support users/devices mobility [5,33]. The process[20] of VM migration includes (i) transferring data from VM's memory to a target physical machine, (ii) sending the state of the resources (CPU, RAM and storage) to a destination node, and (iii) suspending VM on a source node and resuming it on a destination node. This process of VM migration is conducted through the following techniques.

5.2.1. VM migration techniques in edge-cloud context

Fig. 15 depicts different techniques of VM migration [74]. In *cold* migration, a VM initially is stopped, then the whole disk is transferred, and finally the VM is resumed. *Live* migration guarantees the running services on the source node while most of the state of VM is being transferred [9]. Live migration consists of two techniques [9] (see Fig. 15). The *pre-copy* migration initially transfers almost all memory pages (e.g., CPU state, registers and optionally, non-pageable memory) to a destination node before stopping a VM for the final state transfer. This technique iteratively transfers the modified memory pages until for

[20] This process is *stateful* migration, while the process of *stateless* migration includes the start of a new VM on the destination node from scratch and the deletion of that VM from a source node. The stateless migration is simple since nor volatile neither persistent state of a VM should be transferred from source node to destination node.

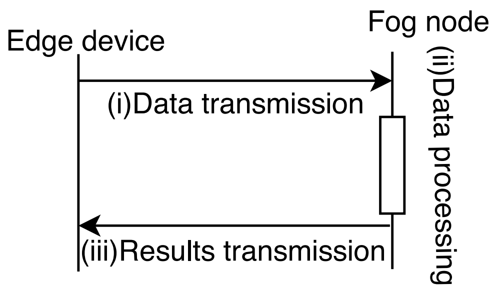


Fig. 14. Execution delay of offloading in edge computing.

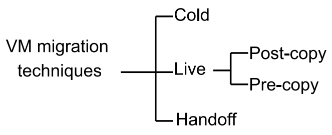


Fig. 15. VM migration techniques.

a pre-defined number of iteration and then stops VM in order to transfer the last modified memory pages. After that, a VM resumes on a destination node. This technique is utilized by hypervisor such as KVM [121], Xen [251], and VMware [74]. A similar approach is reflected to as VM teleportation in VirtualBox [237]. The *post-copy* migration is quietly the opposite of the *pre-copy* migration. Post-copy migration technique stops a VM and transfers the execution of the VM to the destination node so that the VM can resume there. Then, this technique copies the memory pages as page faults happens.

The performance metrics for cold and live migration are *migration* and *down* time [91]. The former is the time to complete migration process, whilst the latter one is the time during which a VM is not running and responsive. Clearly, the down time equals to the migration time due to suspending a VM before its state transfers, making this technique unsuitable to both cloud and fog computing. The down time of pre-copy migration is less than its migration time since a VM is stopped after freezing most of the memory pages. Post-copy migration copies the memory pages once as is done in the cold migration and in the copy phase of the pre-copy migration. However, it incurs shorter down time though its performance is degraded by pages fault, making it unsuitable technique for fog computing. Table 8 compares different migration techniques, where *down* time reduces from cold to live (i.e., first pre-copy then post-copy), and the responsiveness of a VM during migration, conversely, increases from live to cold migration. This is because in cold migration a disk and RAM data are transferred [189], whilst in live migration only RAM data is transferred. Live migration techniques are applicable in the context of cloud computing, especially within datacenters where shared disks are exploited through storage area network (SAN) or network attached storage (NAS) devices [52,167]. Amongst Google, AWS, and Amazon, only Google currently supports live migration.

The migration technique discussed above are no longer efficient for fog computing because they either incurs high down

Table 7
Deployment of IoT applications in the virtualized edge computing.

Work	Virt. type	Mobility	Hardware resources	Solution	Objective functions
[58]	VM Container	No	Intel Atom N2600	Heuristic	Maximizing E2E performance
[270]	VM	No	MEC server	Concrete Heuristic	Minimizing the average of data transfer among fog nodes Minimizing the latency for storing and retrieving data
[14]	NS	No	Small cell	Heuristic	Minimizing the energy efficiency
[265]	NA	No	FN	Heuristic	Minimizing execution time
[147]	VM	No	FN	Heuristic	Reduction in latency
[99]	Container	No	RPi	Heuristic	Maximizing throughput
[205]	Container	Yes	Penguin Relion 1752	Heuristic	Maximizing resource utilization and minimizing latency
[151]	NS	No	FN(300-1400MIPS, 256 MB-2 GB RAM)	Concrete	Maximizing resource utilization in terms of number of services deployed
[213]	NS	No	Sense actuate, and process devices	Heuristic	Maximizing resource utilization
[212]	NS	No	Sense actuate, and process devices	Meta-heuristic	Maximizing resource utilization
[8]	NS	No	Computer desktop, sensors	Concrete	Optimal placement of services in the edge context
[34]	NS	No	FN(2-8 GB RAM, $4-40 \times 1000$ MIPS)	Concrete (ILP)	Minimizing the overall latency
[59]	NS	No	FN(2-16cores CPU, 8 GB RAM)	Heuristic	Minimizing monetary cost, energy and runtime
[79]	NS	No	FN	Heuristic	Minimizing monetary cost of servers and bandwidth through different heuristic algorithms
[156]	NS	No	320 FN	Heuristic	Maximizing resource utilization in term of computing and network
[223]	NS	No	Sensors	Heuristic	Maximizing network utilization
[27]	NS	No	Cloudlet, BS, smart devices	Concrete	Minimizing monetary cost using ILP
[45]	VM	No	1vCPU, 2 GB RAM	Concrete	Minimizing E2E latency using ILP
[246]	NS	No	NA	Approximation	Maximizing resource utilization
[15]	NS	No	RPi	Heuristic	Optimizing E2E latency
[64]	Container	No	ARM A8, 256 MB RAM	Heuristic	Optimizing resource utilization
[258]	VM	No	FN(a quad-core CPU, 4 GB RAM)	Heuristic	Minimizing execution time
[137]	NS	No	NS	Heuristic	Maximizing resource utilization
[19]	VM	No	Server(2,4 cores, 8-64 GB RAM)	Meta-heuristic	Optimizing multi-objective functions in term of cost, latency, user footprint and support
[83]	NS	No	Dual core ARM CPU, 64 GB SD	Meta-heuristic	Minimizing the execution time
[25]	NS	Yes	Server	Heuristic	Minimizing the operation cost
[179]	VM	Yes	Smart phone, small cell	Heuristic	Minimizing the cost placement and migration of application's component
[227]	VM	No	Small cell	Concrete	Maximizing resources utilization in terms of Fog nodes
[97]	NS	No	FN(1-2 GHz, 1-2 MB RAM)	Concrete	Minimizing network cost using ILP
[116]	NS	No	FN	Heuristic	Maximizing the utilization of computing resources
[180]	VM	Yes	MEC server	Heuristic	Minimizing latency under constrained budget
[219]	NS	Yes	Cloudlet	Concrete	Minimizing latency using ILP
[214]	VM	No	Cloudlet	Heuristic	Maximizing the utilization of fog resources
[244]	Container	Yes	FN	Heuristic	Reducing delay of IoT applications
[262]	NS	No	FN	Heuristic	Reducing delay for IoT applications
[228]	NS	No	Desktop computer	Heuristic	Minimizing executing time for mobile services via convex optimization
[4]	VM	Yes	User base	Heuristic	Minimizing the response time and maximizing throughput

Virt: Virtualization, NA: Not Applicable, NS: Not Specified, RPi: Raspberry Pi, BS: Base Station, FN: Fog Node, MIPS: Million instructions per Second, SD: Secure Card, MEC: Mobile Edge Server.
Smart devices include smart phones, tablets, smart glasses, and so on.

Table 8
Comparison among different VM migration techniques.

VM migration technique	Performance metrics	Down time	Type of transmitted data	Context applicability
Cold Migration	Down time	High	Disk	NA
Pre-copy	Down time	Low	Disk, RAM	Cloud
Post-copy	Down time	Very Low	Disk, RAM	Cloud
Handoff	Completion time	Very Low	Overlay contents	Edge

time or degrade response time due to probably frequent pages faults as happens in post-copy migration. Hence, there is a need of virtualization technique(s) with lower down time and more efficient in performance and energy with relatively low resource requirements in hardware. To achieve this purpose, it should be considered several aspects of edge computing that are not valid in cloud computing. (i) Fog nodes span through a WAN, where there is no shared disk and lower bandwidth compared to a LAN within cloud datacenters. Thus, it is effective to minimize transferred data during VM migration. (ii) Most fog nodes share temporal data and typically transfer the analyzed data, and it thus is not supposed to store persistent data in disks. Hence, in edge computing, typically the state of the RAM should be transferred. (iii) Due to running time-critical applications in edge computing, it is essential to reduce the total time required to transfer a VM from a source to a destination node.

VM handoff [204] mitigates the drawbacks of live migration and leverages VM synthesis to seamlessly transfer a VM between two nodes. VM synthesis speeds up VM provisioning in the destination node by dividing VM into a *base* VM (e.g., Linux VM image) that is pre-populated in the destination edge node and the *overlay* VM that is only transferred from source edge node to destination one as needed. Thus, in contrast to cold and live migration, VM handoff reduces the *completion time*[21] defined as the total duration time from the commence of VM hand-off in a source edge node until a VM resume in the destination edge node. This is because the size of overlay VM – the difference between base VM and the desired custom VM [92] – is small. The size of the overlay VM can be further reduced by exploitation of compression and de-duplication techniques in the pipelined stages [91]. However, the completion time of a VM handoff is relatively high and it is required a lighter virtualization technique to be more agile. Ma et al. [132] tailored container virtualization as a complementary technique to a VM handoff by reduction in a file system transfer size in order to accelerate VM migration.[22] To make a VM handoff more effective, it is required to perform optimization in (i) the selection of compression algorithms and de-duplication methods, and (ii) the number of VM overlay creation based on the bandwidth networks between two edge nodes/servers, and availability of computing resources for creating VM overlay in a source edge node [91]. In addition to reduction in the size of VM overlay, multi-path TCP [48,191] is another solution to speed up VM migration, masks the effect of IP changes over WAN, and enhances connection connectivity between two nodes. With respect to VM migration in edge computing, migration *policy* and *strategy* should be considered. Migration policy relates to when a user's VM should be migrated, in which user's speed, direction and position are important. Migration strategy corresponds to where a user's VM should be moved. Shortest distance and lowest congested link between edge devices and fog node are the simple strategy to speed-up VM migration.

[21] Note that the completion time includes down time.
[22] Container Platforms such as Docker and LXC partially or completely support live migration but they cannot be exploited in the edge context due to requiring distributed storage over WAN.

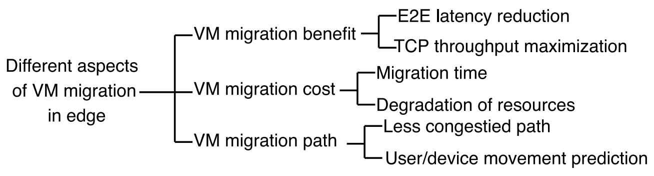


Fig. 16. Different aspects of VM migration in edge computing.

5.2.2. VM migration optimization in edge computing

We discuss VM migration in the following aspects as depicted in Fig. 16.

VM migration benefits. In contrast to the traditional VM migration in cloud to reduce energy consumption and enhance resource utilization, VM migration in the edge context brings the following benefits. The first benefit is *end to end (E2E) latency reduction* which has already been discussed and depicted in Fig. 14. The latency in steps (i) and (iii) can be optimized through VM migration, whilst the one in step (ii) can be optimized through well-allocated VM to task where the availability and capability of resources should be taken into consideration. The second benefit is *TCP throughput maximization* defined as the rate of data successfully delivered over a TCP connection and affected by *TCP windows size* and *round trip latency* factors. The increment in the TCP window size results in higher TCP throughput, which requires more memory for buffering on an edge node. Due to the limited resources in the edge nodes, this solution may not be effective to increase TCP throughput. The second factor can be affected by the physical distance between the edge devices and the edge nodes increases. The more distance, the more deployed network hops to transfer data, which degrades the TCP throughput. However, like E2E latency, TCP throughput can be improved through VM migration from a high to less congested edge node at close distance to edge devices.

VM migration cost. VM migration imposes cost in terms of *migration time* and *degradation of resources* (Network, CPU, RAM and disk) utilization. Migration time is directly affected by the VM migration technique and the bandwidth provisioning between the source and destination edge nodes, and the amount of transferred data (i.e., memory and disk state). As an example, in pre-copying live migration, the amount of data is the memory size of a VM and the average number of dirty memory page yielded in each time slot. The second dimension of the VM migration cost is degrading the resource utilization because VM migration acts as an I/O operation. Thus, VM migration degenerates I/O applications and it should be performed on the edge nodes that process computing-intensive applications or I/O applications with low I/O footprints.

VM migration path. As discussed, VM migration is a viable solution to react to device mobility in order to reduce E2E latency and improve TCP throughput. However, VM migration is not an effective way in some cases when the migration cost outweighs the migration gain. Instead, providing a new optimized path with less congested traffic [163] between the edge node and VM deployed in the edge nodes to send and receive data can be an efficient way from performance perspective [30,187]. Thus, VM migration along with an optimized path between edge nodes and

Table 9
VM migration in the edge context.

Work	Virt. type	Virt. techniques	Hardware resources	Objective/Cost
Puliafito et al. [189]	Container	Cold live	RPi3	(i) Evaluation of throughput and RTT (ii) Evaluation of down time, page faults and transferred data
Kakakhel et al. [108]	Container	Cold live	Cloudlet (1 vCore, 1 GB RAM)	(i) Evaluation of throughput and RTT (ii) Evaluation of migration, down time and pages fault
Bittencourt et al. [37]	VM Container	Handoff	Cloudlet	(i) Propose a general architecture to support VM migration (ii) NA
Saurez et al. [205]	Container	Live	Fog node (6 cores, 48 GB RAM)	(i) Proposing migration APIs, discovery resources (ii) Migration time, discovery time for fog nodes
Tang et al. [224]	VM Container	Handoff	Fog Node	(i)Latency and power consumption reduction (ii) Migration time
Lopes and Higashino [129]	VM Container	Live Handoff	Fog nodes	(i) Proposing a simulator for VM migration in FC (ii) Migration time measurement for VM/container in FC
Qiu et al. [191]	Container LXC	Live	Cloudlet 1 vCPU, 1 GB RAM	(i)Reduction in migration time and improve resilience of migration process via multi-path TCP
Filiposka et al. [73]	VM	Handoff	Cloudlet (1–2 cores, 1-2 GB RAM)	(i) A simulation-based evaluation for VM migration for mobile users with speed of 1-2mh (ii) NA
Gonalves et al. [87]	VM	Proactive	Cloudlet (2800 mi, 8 GB RAM)	(i) A simulation based VM migration to reduce migration times and unavailability VM (ii) Reduction in migration time
Zhang et al. [268]	VM	Live (pre-copy)	NA	(i) A numerical evaluation of reduction in network traffic via VM migration for mobile users (ii) Migration time
Majeed et al. [2]	Container	Live (Stateless)	Cloudlet (2cCPU, 64GB RAM)	(i) Investigation of container-based offloading across devices, fog nodes, and cloud based on ML approaches using resources utilization (ii) NA
Chaufournier et al. [48]	VM	Live (Pre-copy)	Cloudlet (8core, 64 GB RAM)	(i) VM migration improvements in terms of throughput, migration time using multi-path TCP (ii) Migration time
Ma et al. [132]	Container	Handoff	Cloudlet (3vCore 16 GB RAM)	(i) Reducing in completion time of container migration via potential reduction in file system transfer size (ii) Down time, transferred data and page faults
Teka et al. [226]	VM	Live	Cloudlet (7cores, 16 GB RAM)	(i) Using multi-path TCP to reduce TCP connection establishment for VM migration, and RTT and throughput measurement (ii) Migration time and down time
Machen et al. [134]	VM Container (LXC)	Live	MEC server (2vCore, 16 GB RAM)	(i) A comparison between KVM and LXC in migration time, down time for several applications (ii) Migration time

fog nodes can improve the target performance metrics. This goal can be further enhanced through finding an optimized VM migration path based on the users/devices movement prediction [73]. Table 9 summarizes the exploited virtualization techniques in the context of edge computing.

6. Virtualization of network resource

Due to the highly dynamic feature of edge paradigms, providing network functions (e.g., firewalls, caches, proxies, intrusion detectors and WAN accelerators) through adding, removing and upgrading network hardware is a tedious task and imposes heavy operational (OPEX) and capital (CAPEX) expenditure costs [39]. Network Function Virtualization (NFV) [153,154] solves this issue by moving network functions from network hardware appliances to software hosted in network devices and servers as Virtual Network Functions (VNFs). VNFs execute a set of network functions ranging from security (firewalls, intrusion detection systems), performance improvements (caches, proxies, traffic accelerators), traffic shaping (load balancer, rate limiters), to name a few. *Connectivity*, *control* and *management* of VNFs is in charge of Software Defined Network (SDN), which shapes network traffic without dealing with network hardware by introducing separation between *control plane* and *data plane*.[23] The control plane software runs on either standard server or network devices and handles packets received by the network devices based on the network rules and policies. The data plane, represented by virtualized hardware (i.e., VNF), forwards and receives packets based on the mandated rules by the control plane. Thus, NFV and SDN are complementary technologies, where NFV provides basic networking functions through hardware virtualization and SDN manages them for specific use programmatically defined and modified. It should be noted that SDN can be deployed in purely hardware network devices (e.g., switches) to flexibly control network flow [206]. However, the core similarity of SDN and NFV is to abstract network functions on the beneath network hardware resources for network applications as illustrated in Fig. 17. SDN abstracts the separation of control data and forwarding data, and NFV abstracts the data plane from the hardware infrastructure. The difference between SDN and NFV returns to how they separate functions and abstract resources. SDN abstracts networking

[23] In the traditional network, IP layer vertically integrates both planes in network devices.

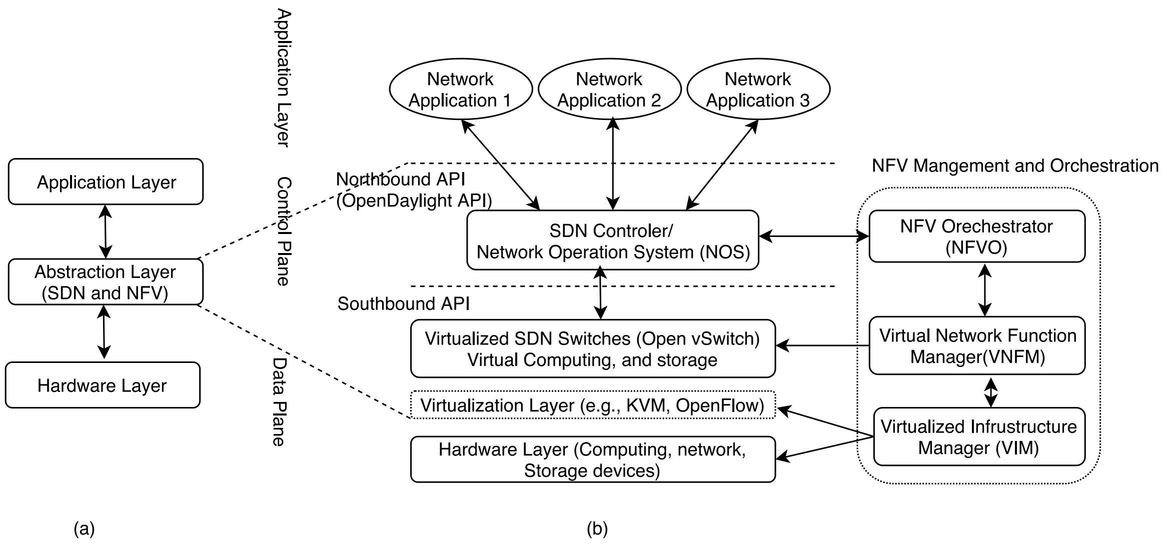

Application Layer
Abstraction Layer (SDN and NFV)
Hardware Layer
(a)
Application Layer
Control Plane
Data Plane
Network Application 1
Network Application 2
Network Application 3
Northbound API (OpenDaylight API)
SDN Controler/ Network Operation System (NOS)
Southbound API
Virtualized SDN Switches (Open vSwitch) Virtual Computing, and storage
Virtualization Layer (e.g., KVM, OpenFlow)
Hardware Layer (Computing, network, Storage devices)
NFV Mangement and Orchestration
NFV Orechestrator (NFVO)
Virtual Network Function Manager(VNFM)
Virtualized Infrustructure Manager (VIM)
(b)

Fig. 17. Architecture of SDN and NFV integration.

resources (e.g., switches or routers) through moving control decision to a virtual network control plane to define source and destination of network flows, whilst NFV virtualizes physical resources beneath a hypervisor to expand/shrink a network without adding/removing network devices. In the following, we discuss integration of SDN and NFV.

6.1. Architecture of SDN and NFV integration

SDN and NFV integration as an*abstract* layer sits between *application layer* and *hardware layer* and consists of the following components as depicted in Fig. 17.

Control Plane: SDN controller/NOS provides a centralized view of a network to steer the network flow/traffic by using OpenDaylight [177] software that exposes northbound API to the network applications to collect and analyze information about the requirements of the network applications. OpenDaylight supports different southbound protocols such as OpenFlow [148], OVSDB [176], and Network Configuration Protocol (NETCONF) [69] to allow connection between SDN controller and data plane (i.e., virtualized switch (OpenVswitch (OVS) [176]) to steer network flow/traffic.

Data Plane: This component consists of virtualized SDN switches (Open vSwitch) and *Virtual Network Function* (VNF) that includes firewalls, domain name system (DNS), caching or network address translation (NAT) on VMs instead of carrying out by proprietary or dedicated hardware. These VNFs run standard server, switches, storage devices or even cloud computing infrastructure with the help of Xen, KVM, VMware hypervisor for computing resources and VXLAN [239] and NVGRE [173] for networking resources.

NFV management and orchestration: This component includes the following sub-components in Fig. 17. (i) virtualized infrastructure management (VIM) orchestrates allocation, deallocation, upgrade of NFV infrastructure and optimization of their use. (ii) Virtual Network Function Manager (VNFM) manages instantiation, configuration, start, stop, scaling in/out, updating, upgrading, suspension, termination of VNFs in Point of Presence (PoP) infrastructure. (iii) NFV orchestrator (NFVO) serves *resource and service orchestration*. The former orchestration provides coordination, authorization, release and engage of NFVI resources (e.g., a fixed set of virtual CPU, a certain amount of memory and disk space) within PoP infrastructure for a given VNF. The latter one manages the relationship between VNFs and provides on-fly scaling in/out, terminating, updating, upgrading and managing of network services topology. Nokia as one of the leading vendor offers CloudBand Application Manager (CBAM) [138] as VNF Manager and CloudBand Network Director as NFV orchestrator. A list of open source projects implementing ESTI NFV MANO framework is in [150].

Network Application: This includes *service/virtual functions* (SFc/VFs) (e.g., firewalls, deep packet inspections (DPIs) and virus scanners in the context of *network protection* system [35]) that defines a specific treatment to the received packet and runs on the virtualized hardware (i.e., VNF creation). A *partially* or *totally* ordered of SFs (called *service function chain (SFC)*) must be applied to traffic network to achieve the required network services/policies. A network policy, as a flow, mandates packets to be traversed according to their associated SFC. For example, a network protection application as a network service, might provide three service functions like firewall, DPI and virus scanner as a SFC. In summary, SCF abstracts a network service in terms of the required SFs and the order in which they should be executed. Virtualization at network level, discussed above, reduces *operational and expenditure cost* via multi-tenancy, and makes flexible, fast and elastic demands of network services through on the fly adding, updating and removing virtual functions without dealing with network hardware appliances. These advantages are necessities for IoT applications that support context aware, ultra reliable and user specific network services. This mandates a well-defined *SDN and NFV architectural design*, well selected *virtualization types* hosting VNF and *performance-oriented optimization algorithms* for VNF management. The first requirement relates to NFV MANO and readers are referred to [39] and the last two requirements we discuss in the following sub-sections.

6.2. NFV virtualization techniques

Virtual Network Function (VNF) executes a particular computing- or bandwidth-intensive network function. VNFs are mainly distinguished in the *underlying hardware virtualization* and the *network functions* they perform. They generally exploit three virtualization techniques as shown in Fig. 18.

Hypervisor based VNF runs on the richness capable hardware and Cloud4NFV [216] and OPNFV [178] platforms, for example, move both data and control planes to the virtual resources. This technique is high in resource consumption and

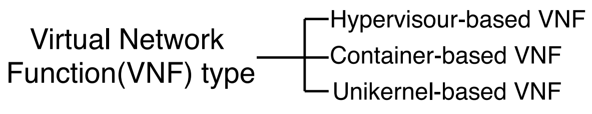


Fig. 18. Different types of VNF.

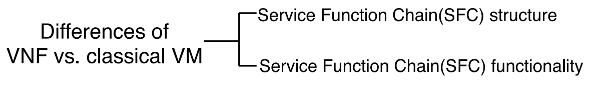

Differences of VNF vs. classical VM
Service Function Chain(SFC) structure
Service Function Chain(SFC) functionality

Fig. 19. Differential characteristics between VNF and classical VM.

slow in movement and deployment. To combat such limitation, **Container-based VNF** allows a network function to be packaged within a lightweight container engine, e.g., Docker. This makes container-based NFV frameworks (e.g., Glasgow Network Functions (GNF) [56]) to be agile in movement and deployment. **Unikernel-based VNF** is a specialized hypervisor based VM and optimized for middlebox processing to execute specific network functions. ClickOS [145], as an example, incurs extensive changes in Xen's I/O to significantly accelerate networking in the middleboxes running in VMs. This design requires several performance related features to be considered. It should support *flexibility* to run network functions on different operating systems, provide *isolation* in terms of performance (CPU and memory) and security to achieve multi-tenancy and guarantee *scalability* in the exploitation of the resources proportional to the workload.

Selection of *x*-based VNF (*x* represents different virtualization techniques) is not a black and white choice since each virtualization technique is the best in some quantitative features, not all (see Fig. 10). Unikernel-based virtualization is the best in *instantiation time* and *image size*, whilst container-based virtualization outperforms other techniques in *memory usage* [56]. *Network throughput*, the rate of successful delivery over a communication channel, is another quantitative feature in which unikernel performs best [145]. *Idel RTT* is the delay that arises from virtualization layer, where hypervisor based VMs act the worst mainly due to copying the packet from hypervisor to VMs; unikernels are the best if packets forwarding is optimized [197]. Furthermore, other features listed in Table 10 should be considered in the deployment of x-based VNF ClickOS and the art of NFV. Hypervisor based virtualization provides *flexibility* to run different guest OS on the same platform in the cost of direct access VMs to NIC devices. This thus complicates live VNF migration. In contrast, containers are inflexible because they run on the same OS, imposing restriction on network operators to run them from different vendors. Hypervisor based virtualization requires alignment of dependencies between guest OS and inter VM networking solution, while containers and unikernels allow running the VNF process directly in a host OS. Thus, unikernels and containers are the best in *service agility*. Open source communities support hypervisor- and container-based virtualization well, while unikernels do not. Hypervisor-based virtualization has the highest degree in *application compatibility*, while container-based virtualization has compatibility in Linux, windows and Mac. Table 10 summarizes the features of different *x*-based VNF, where the higher rank indicates better performance in the associated feature. For example, unikernel with rank 1 is better than hypervisor-based VM with rank 3 in terms of *instantiation time*.

6.3. VNF management in cloud- VS. edge-based paradigms

In addition to the selection of virtualization technique, VNF orchestration and management is another imperative issue in NFV-enabled networks. We investigate it from the following perspective.

VNF vs. classical VM. As illustrated in Fig. 19, VNFs and classical VMs have the following essential differences. **First**, VNFs must run task in a sequence in the form of SFC, while classical VMs do not. A SFC determines a partially/totally ordered sequence for network traffic/flow that must be traversed through VNFs. This ordered sequence can be *liner*- or *graph-based* shapes [35]. In the liner shape, all service functions must be serially conducted, while in the graph based shape they can be executed in parallel as long as the sequence among service functions in SFC is preserved. SFC shapes can be either *static*, or *dynamic* in which the number and shape of service functions vary with time and load [35]. The dynamism of the SFC shape makes indispensable virtualization of service functions in network appliances. **Second**, VNFs are differentiated in functionality, the required resources and their interrelationships, whilst classical VMs are usually the same in functionality. For example, firewall and NAT functions need lightweight resources, while IDS function requires intensive computing resources. This makes a challenging issue in the synergy of cloud and edge paradigms due to leveraging devices with a variety of resource capabilities. Therefore, due to these important differences, the designed algorithms for classical VM placement and management are not applicable in the NFV enabled networks and revisiting such algorithms is a must.

VNF Placement in cloud- vs. edge-based: As shown in Fig. 20, the following factors impact the management and placement of VNF in the edge-cloud environment.

(1) *Resource richness:* Due to resource constrained edge devices, essential factors for VNF deployment in PoP locations are *instantiation time*, *memory usage*, and *idle delay* [56]. These requirements can be met through container based virtualization. It is noteworthy that unikernels cannot be deployed in edge devices (e.g., customer premises equipment and home routers) lacking support of hardware virtualization since unikernels are built and modified on top of hypervisor. Furthermore, this resource constraint results in different objective functions and QoS constraints for edge paradigms compared to the ones for cloud paradigms as discussed in the next section.

(2) *Network/substrate topology:* This refers to an organization of hardware servers and their connection through physical links. Network topology and bandwidth are two essential aspects of differences between two paradigms. Cloud-based datacenters follow a centralized and well-defined network topology (e.g., fat-tree [10]) with high bandwidth connection between servers, while edge-based paradigms usually exploit irregular topology with constrained bandwidth. Thus, computation optimization is more important than the bandwidth optimization for cloud-based paradigms, while these two optimization problems become an imperative trade-off for edge-based paradigms in the context of VNF placement. This results in expanding/compressing SFs across servers within cloud datacenters/edge computing devices. This is because the more SFC is split, the need for high richness server decreases and the demand for bandwidth increases.

(3) *User/device mobility:* Shifting from a host-centric to a data-centric model provides computational resources to end users to support device/user mobility. MEC, as an example of edge

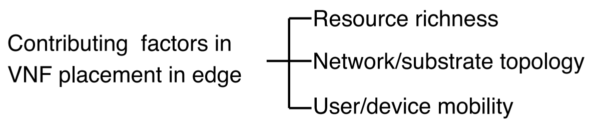

Contributing factors in VNF placement in edge
Resource richness
Network/substrate topology
User/device mobility

Fig. 20. Differential characteristics for VNF placement in cloud and edge paradigms.

Table 10
Comparison between different *x*-based VNF (*x* is hypervisor, container and unikernels) in quantitative features (i.e., the first five features — instantiation time, image size, …, idle RTT) and non-quantitative features (i.e., flexibility, …, application capability). The higher the rank, the better *x*-based VNF in terms of features.

Features	Instantiation time	Image size	Memory usage	Network throughput	Idel RTT	Flexibility	Service agility	Management framework	Application compatibility
Rank 1	Unikernel	Unikernel	Container	Container	Container	Unikernel	Container	Hypervisor	Hypervisor
Rank 2	Container	Container	Unikernel	Unikernel	UNikernel	Hypervisor	Unikernel	Container	Container
Rank 3	Hypervisor	Hypervisor	Hypervisor	Hypervisor	Hypervisor	Container	Hypervisor	Unikernel	Unikernel

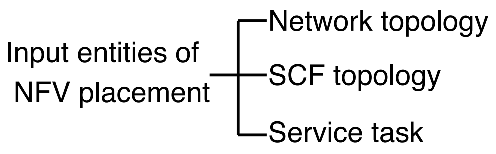


Fig. 21. Input entities of VNF placement.

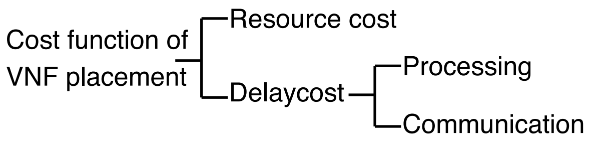


Fig. 22. Cost function of VNF placement.

paradigms providing such model, supports context-aware network services for which VNF migration is a necessity to guarantee the required QoS. In contrast, the cloud-based paradigms do not support such applications and VNF migration thus does not happen. In both cloud and edge paradigms, VNF migration might happen in order to reduce operational cost and maintain acceptable end-to-end flow performance as discussed later.

Therefore, as mentioned above, the synergy of cloud and edge computing creates an extra level of complexity for VNF management across distributed resources with a variety of capabilities. A key challenge relating to this issue is how to make a balance approach in resource utilization of both edge- and cloud-based paradigms for NFV placement. As an example, VNF deployment in edge computing improves response time while reduces availability, and vice versa as if it is located in cloud computing. *Where* to place VNFs for a given set of PoP locations and *how* to steer network traffics across them (i.e., SFC provisioning) mandate to specifically define objective functions and QoS requirements.

6.4. VNF placement and migration in edge-cloud environments

NFV provides network services in the form of a software-based virtualization entity as VNF. VNFs are logically connected and executed based on SFCs (represented user's requests). One essential issue relating to VNFs and SFCs is how to manage them in an edge-cloud environment so that performance and cost are optimized while the required SLAs for SFCs are preserved. This issue arises four questions. (i) Where to initiate a VNF for a given PoP locations across an edge-cloud environment? (ii) How many resources such as CPU, network bandwidth, RAM capacity, and probably storage space should be allocated to a VNF? (iii) How to steer SFCs among determined VNFs based on the predefined order between SFCs? (iv) When VNF should be migrated and consolidated in reaction to changes in user's requests (SFCs) and resources statues? The first three questions define a *static* VNF placement problem and the last one results in a *dynamic* one.

Input entities of VNF placement: To solve static and dynamic VNF placement problems, three input entities, as shown in Fig. 21, should be taken into consideration. (1) *Network topology:* The edge-cloud environment has a hierarchical or graph-based architecture, where lower layers are located at the proximity of users with limited bandwidth while higher layers are located far away from users with higher bandwidth. (2) *SCF topology:* It can be a *simple line*, or *bifurcated* in the real scenario for network applications [35]. This creates a critical issue in how to split SFC topology to steer data flow between VNFs. Obviously, this issue can be eliminated if the whole of a SFC is located in a physical server. (3) *Service tasks:* They are different in requiring resources allocated to VNFs based on their functionality. For example, NAT, as a service task, requires light-weight computing resource, but quick in response since it remaps one IP address into another by changing network address information in the IP header of packet. While an IDS as a service task needs intensive-computing resources due to processing data with high rate.

Cost function of VNF Placement: With respect to the input entities, VNF placement optimization consists of costs depicted in Fig. 22. (1) *Resource cost* can be translated into the amount of computational and communication resources used, the number of servers deployed, or monetary cost if it can be applicable to resources usage. (2) *Delay* is the overall *processing* and *communication* [124,152] time to process a service task. Processing time includes the time needed for the VM hosting VNF to apply network operations on the arriving packet and depends on the capacity of computational server/device. The communication time consists of *transmission* and *propagation* time. The former is the time to transmit a packet over a link and depends on the packet size and link bandwidth. The latter is the time to be required for transmitting signal data from the source to destination nodes and measured based on the distance only. Propagation time can be ignored if the packet size is large. All these delays in all layers of an edge-cloud environment should be jointly optimized especially for URLLC services.

Considering input entities and cost functions discussed above, two optimization problems can be defined. (1) The first one is resource usage and end-to-end latency of service chain (simply response time) optimization problem along with a QoS like a specific rate of SLA violation. This optimization problem can be reduced to the conflicting optimization problems due to different capabilities of resources spread across the edge-cloud environments. As an example, a trade-off between response time and availability, where an VNF deployed in the edge computing resulting in low latency, and in the cloud leading to high availability. Thus, it is not possible to optimize both latency and availability through VNF deployment either only in edge or only in cloud computing. (2) The second optimization problem is to make decisions on *when* and *where* network traffic should be steered. This optimization problem is formulated in the form of Integer Linear Programming (ILP) problems [70,254,266]. They have high complexity in computational time, making them an unsuitable solution for large scale edge-cloud environments. To tackle such problem, most researches make effort to propose greedy solutions to provide fast decision on VNF placement and management [195,196].

Due to the dynamism of edge-cloud environment and changes to SFCs, the initial decision on VNF placement cannot be well cost-efficient. Hence, it is necessary to re-optimize the optimization problem in each time slot, periodical time, or based on the unsatisfactory in a specific SAL (e.g., SFCs rejection lower than a specific percentage). To cope with this issue, it is required to design algorithms for *horizontal* and *vertical* scaling of VNFs [230]. A VNF is vertically scaled up through allocation of more resources, and scaled down through releasing redundant resources. A VNF is horizontally scaled up/down through adding/removing resources. VNF migration is orthogonal to scaling down/up and creates reconfiguration costs for VNF movement between physical servers and re-steering SFCs among VNFs. The reconfiguration degrades response time and causes temporal service interruption and thus a dynamic VNF placement requires to carefully consider the migration cost of VNFs. Table 11 maps the provided taxonomy to the state-of-the-art studies in networking virtualization.

7. Future research directions

Based on this review, this section identifies a few key research directions in virtualization of computing and network resources in edge-cloud environment.

Selection and Deployment of Computing Resources Virtualization: Hypervisor-based, container-based, and unikernels are the three main virtualization techniques [225]. The selection of one of these techniques and making it adapt to different use-case scenarios in the context of edge computing is not an easy task. We should take three factors into consideration. The first factor is the *virtualization properties* such as instantiation time, image size, memory footprint and live migration support. The second factor is the *requirements of IOT applications* [165] mainly response time, mobility, scalability and multi-tenancy. The last factor is the *capability of the resources* [261] in terms of memory, storage, computing and networks. A wise consideration of these factors can help a well-designed and deployment of virtualization techniques although we should compromise between, for example, security of hypervisor-based and agility of container-based virtualization.

Mobility Services in Edge Computing Paradigms: Beyond ultra-low latency and high bandwidth demands, *mobility* is the most distinction factor in resource management in edge computing compared to cloud computing. Mobility of devices either being transported by itself, human or another carrier, is one of the essential features that influences the selection and deployment of lightweight virtualization techniques (i.e., unikernel and container) which are more agile [135,208]. The deployment of such lightweight techniques cannot satisfy users/devices with high speed (e.g., cars and trains), and *user movements pattern* and *acceleration of VM transmission* should be considered. We classify user movements into *planed* (i.e., social events), *regular-based* (i.e., workplace) and *unpredictable* movements. For the planned movement, fog nodes can simply be provided in the event placement to serve users (e.g., sports stadiums) [128]. For the regular-based movement, the historical and statistical prediction of user/device movements [86,164] leveraging ML or other optimization approaches, e.g., Bayesian, can be exploited. For the last movement type, cumulative distribution patterns of users' movements is effective to find proper paths for a large number of users rather than for a particular user. In respect to VM transmission, the VM overlay technique [91,92] using effective compression algorithms is a way to migrate VM for entities with high speed. Overlay VM is the difference between a base VM and the current deployed VM.

Most of the reviewed studies, as listed in Tables 9 and 11, consider only the mobility of devices/users for VM placement and migration. This is valid in the context of MEC due to the fixed MEC servers, while in fog computing the location of both edge devices and fog nodes can change over time [264]. This makes VM placement and migration more challenging because the resource discovery is required and VM migration happens not only due to performance degradation but also because of disappearing a fog node serving mobile devices. Thus, it is required to design mobility-aware methods/algorithms for offloading the tasks from mobile devices to VMs in MEC servers or in fog nodes based on network coverage, network load and the acceptable execution delay.

Virtual Network Function (VNF) Placement and Deployment: The emergence of SDN and NFV decouples hardware properties from network functionality (e.g., firewall, proxy server, and IDS) and deploys them upon generic computing resources under a central control [39]. SDN-/NFV-enabled network reduces monetary cost and improves E2E latency in the edge computing paradigms [256]. Nevertheless, the deployment of VNFs increases the utilization of resources in a central cloud and reduces E2E latency in the edge computing. To make a balance, it is required to make decisions on *where* (edge or central cloud) to create VNF and *which* VNF should serve the requests/service functions so that applications receive acceptable E2E response time and service functions violation. One solution for such balance can be the binary selection of either edge or cloud computing to serve a SFC, which suits to the two layered edge-cloud computing architecture (e.g., MEC).

Nonetheless, for the multi-layer edge-cloud architecture (e.g., FC), deployment VNFs and steering SFCs among servers is more challenging issue in comparison to the ones in the two layered architecture. Steering SFCs across servers in these architectures can be considered as a spectrum. From one extreme side of spectrum, the whole SFC is put in a server to achieve high utilization of resources while increases response time, and from another extreme side, a SFC can be spread across servers that raises bandwidth consumption and reduces response time. Hence, there is a need of a balance between these two sides based on the type of applications. *Real-time* applications are sensitive to response time and it is a need of compact SFCs in a server. For *semi-/non-real time* applications, users have more freedom to scatter SFCs across servers. Furthermore, behaviors of users have a determinative rule in VNFs placement, where a user's service consumption and mobility respectively specify which server should host the VNF and when the VNF should be migrated from one server to another. The network dynamism is another factor that may influence the QoS requirements in the long term, which necessitates to recompute VNF placement and migration. Optimization of VNF placement and migration along with steering SFCs can be formulated in Integer Linear Programming (ILP) problem, which yields high time complexity [41,269]. Thus, the exploitation of efficient online greedy algorithms is an effective solution to cope with such complexity, which requires theoretical and experimental evaluations.

VNF Virtualization and Management: The existing research on the NFV framework in the edge-cloud environment relates to VNFs placement with SLA satisfaction and NFV/SDN architecture for Wi-Fi networks, wireless networks and Virtualized Customers Premises Equipment (vCPE) [39]. However, virtualization generally degrades the performance of a system and thus there is a venue to evaluate the impact of virtualization on VNF performance in terms of process sharing, TCP/UDP throughput, packet loss and packet delay [209]. Furthermore, complications arise when multiple different service providers cooperation is required. Traditional NFV and SDN approaches are suitable for single-owner networks and data centers where all network routes and devices are controlled by the same entity. In contrast, routing network packets through a third-party network may be complicated or unacceptable due to trust and performance issues. Therefore, flexible and secure cross-network resource hand-off protocols and software implementations must be developed in order to support dynamic service migration for highly mobile clients. Lastly, implementing a virtualized edge infrastructure

Table 11
Comparison of VNF deployment in edge-cloud environment.

Work	Domain	Virt. type	NFV OP	SFC	Cost function	Solution
J. Son, R. Buyya [218]	Edge-cloud	NS	NFV-P	Yes	E2E latency	Sim.- Heuristic
Toosi et al. [230]	Cloud	VM	NFV-M	Yes	Resource cost	Sim.- Heuristic
Bhamare et al. [36]	Edge-cloud	VM	NFV-P	Yes	Delay cost	Imp.(EC2)- ILP and Heuristic
Li et al. [125]	Edge-cloud	NA	NFV-P	Yes	Resource cost	Sim.- ILP and Heuristic
Xie et al. [252]	Edge	Container	NFV-P	Yes	Resource cost(bandwidth)	Sim. - ILP and Online
Ghaznavi et al. [80]	Cloud	Unikernel	NFV-P	Yes	Resource Cost(CPU)	Sim. - ILP and Heuristic
Brogi et al. [41]	Edge-cloud	NA	NFV-P	Yes	Resource/Delay costs	Sim. - Heuristic
Wang et al. [240]	Edge	NA	NFV-P	Yes	Resource-cost	Sim. - ILP and Hungarian
Yang et al. [256]	Edge	VM	NFV-P	Yes	Resource-cost	Sim. - Online
Drxler et al.[66]	All	NS	NFV-P	Yes	Resource-cost	Sim. -MILP-Heuristic
Cziva et al. [55]	Edge-cloud	NS	NFV-P/M	Yes	E2E latency	Sim. - Optimal stopping theory [185]
Chen et al. [51]	Edge	NS	NFV-P	No	Resource-cost	Sim- Genetic and Heuristic
Chemodanov et al. [50]	Edge-cloud	NA	VNF-P/M	Yes	Resource cost(bandwidth)	Sim. - ILP and Heuristic
Zhang et al. [267]	Edge-cloud	NA	NFV-P	Yes	Resource cost	Sim. - ILP and Heuristic
Nguyen et al. [169]	Edge-cloud	NA	NFV-P	No	Resource-cost	Sim. - non-convex IP and Markov approximation
Zhang et al. [269]	Edge-cloud	NA	NFV-P	No	Resource/Delay costs	Sim - Heuristic
Santa et al. [203]	Edge	VM	NFV-M	No	NA	Impl.
Behravesh et al. [31]	Edge-cloud	NA	NFV-P	Yes	Resource cost	Sim. - MILP
Ren et al. [195]	Edge	VM	NFV-P	Yes	Resource/Delay costs	Imp. - Approximate and Heuristic

Virt: Virtualization, OP: Operation (P: Placement, M: Migration), NA: Not Applicable, Sim: Simulation, Impl: Implementation.

hosted by telecommunication providers that is shared between multiple service providers on demand would be potentially useful. Such infrastructure can enable moving services closer to a large number of clients based on the current or the predicted load. Modern virtualization techniques are highly suitable to this type of resource sharing between multiple independent tenants.

Synergy of IoT, AI, Blockchain, and Edge Computing: The impact of emerging technologies (IoT, AI, and Blockchain) are essential on the computing at the network edge in the case of data processing, data storage and management. IoT transfers a world of connected devices and servers into a world of data in the context of edge and fog computing. Artificial Intelligence (AI) allows to process data at the edge of network to make real-time actions without processing data at the core network (cloud servers). This reduces data transfer to the cloud and provides better response time. Blockchain records data in a distributed ledger/database. Due to distributed and resource-constrained devices at the edge network, the impact of these technologies is different with the one on the cloud computing investigated by [84]. The connectivity and richness of IoT devices and fog servers affect the AI (Machine Learning (ML) + Deep Learning (DL)) and blockchain at the edge computing. Since the model training of AI is computing-intensive, we need to deploy lightweight model training or move the processing into fog servers instead of edge devices. Another solution is to leverage distributed and federated model training across edge devices [247]. This solution, however, brings communication overhead, where the edge computing faces with limited bandwidth. In respect to each of these solutions, it might be effective to reduce dimension of data with the help of AI [82] since computation time of a model training is directly proportional to the data dimension and instances/records. With these steps, we approach to have data analytics at the edge computing as discussed more later. Integrating blockchain and edge computing enhances integrity, security, privacy, processing, tracking and monitoring of data via peer-to-peer network with the help of consensus mechanism such as Prof of Work (PoW) and Prof of Stack (PoS) network [57]. Nevertheless, this technology is not simply to integrate with edge computing due to heavy processing for consensus protocol and storing a copy of each block on all devices [130]. Moreover, this technology needs more energy and bandwidth [120]. Thus, to take the advantage of blockchain, it is required to implement a lightweight blockchain across edge devices and edge servers so that distributed ledgers are placed in the more powerful servers in terms of storage. Private and public blockchain, respectively running on a local subnet and across subnets of edge computing, is another solution to reduce storage and bandwidth consumption.

Synergy of Virtualization, Machine Learning, and IoT Data Analytic: Due to the massive amount of data at the edge computing, centralized data processing in a cloud computing infrastructure raises several interesting issues in terms of high bandwidth demand and high latency [271]. However, the decentralized data processing in the edge computing with the help of ML techniques copes with such issues [3]. We can well imagine (i) IoT big data is processed only in a cloud or edge, which respectively suits to delay tolerant and delay sensitive applications, and (ii) both cloud and edge computing are jointly exploited to analyze IoT big data. ML and IoT data analytics meet management and control of virtualized resources in order to have their global status [207]. VIM, MANO and SDN, already discussed, expose such status and control the available virtual resources for IoT big data analytic in the cloud-edge environment. ML techniques can leverage randomized and distributed algorithms [207] to combat the high dimensional big data in terms of features and samples number generated by IoT devices. Randomized algorithms are not suitable to IoT platforms since these algorithms iteratively run on centralized data. However, distributed algorithms enable the processing of data locally in each computing devices/node in an edge; the results of the locally processed data contribute to global decision making. This arrangement increases the communication between nodes for distributed algorithms deployment and management. To enable IoT big data analytic in a cloud-edge platform, it would be effective to (i) map different stages of big data analytic (data acquisition, feature extraction, models processing, uploading and feedback and information extraction [273]) to cloud-edge layers based on the richness of resources, (ii) design distributed algorithms with low communication requirements and convergence in a reasonable iterations and (iii) leverage algorithms that enable the utilization of online sample data instead of offline. Online distributed algorithms are thus the best fit in the context of IoT big data analytic because online and distributed features of algorithms respectively allow to process the new arrival of the data generated by edge devices and analyze data in itself constrained-resource devices.

Data Reduction Techniques in Edge Computing: Due to resource-constrained edge devices for processing the expected huge amount of IoT data, data reduction/cleaning is an essential process to reduce storage cost, decrease energy consumption, prevent I/O and bandwidth bottleneck. Data cleaning, namely, is to remove missing values, redundant values and outliers so that data integrity and reliability are maintained. In addition, data can be reduced with simple, but very effective approaches, such as sampling, filtering, summarizing, and compression [103]. However, these typical data reduction approaches directly are not applicable in the edge environment due to distribution nature of resources and online/time-series data generation. Thus, low-complex distributed data cleaning algorithms would be appropriate for such environment in which the generated data are transferred to the edge/fog severs to apply different data reduction techniques [243]. These techniques can be raw data reduction via Pearson coefficient metrics, filter data redundancy via K-nearest neighbor clustering, and more complicated and effective approaches [182,236]. To cope with the processing of online data generated in IoT devices, it is essential to apply data reduction approaches with least delay and re-configuration. Traditional approaches such as aggregation techniques (e.g., summarizing) and event/policy-based events are not suitable for reducing online data. This is because these approaches require a posterior dataset or need an engine to pass specific data without actual data reduction [181]. To figure out this issue, the effective way is to streamify data with the help of caching and Perceptually Important Points (PIP) techniques in which the accuracy of forwarded data and forwarding delay should be balanced [181]. Solely relaying on indicated solutions to address the reduction of distributed and online data in edge computing might not be efficient. Instead, it is effective to leverage centralized data reduction approaches [210] in the edge devices to remove dirty data, and then the cleaned data should be transferred into the edge servers to apply ML and DL to reduce dimensions and instances of data [82]. Thus, a pipeline data reduction approach should be designed throughout the edge-cloud ecosystem to achieve raw data reduction, packet reduction, and event reduction [96].

8. Conclusions

The edge computing paradigms move cloud-based services closer to the edge of network hosting IoT applications requiring ultra-low latency and jitter, mobility and location awareness services and high demand bandwidth. The edge computing paradigms consist of a variety of resources with different capabilities in richness, where virtualization is the essence of their infrastructure. In this paper, we first analyzed different paradigms of cloud and edge computing and investigated the reasons behind transition from one paradigm to another based on the requirements of IoT applications. More specifically, we have identified the objectives, features and pillars for both edge and cloud paradigms to understand which paradigm is more suitable to a particular IoT application. Second, we discussed different virtualization techniques for computing and networking resources and explained which virtualization technique could be appropriate based on the richness of resources and the requirements of IoT applications. We then investigated the classical VMs and Virtual Network Function (VNF) from performance perspective in an edge-cloud environment. We finally map the existing studies to the provided taxonomy for each research subject in this paper. Based on the lessons learned from the reviewed studied, we have identified the future research directions and the potential starting point to tackle the challenges regarding virtualization of edge computing paradigms. The challenges include the determination of virtualization type, placement and migration virtualized computing and network resources in the edge-cloud environment adapted to the requirements of IoT applications deployment.

CRediT authorship contribution statement

Yaser Mansouri: wrote the paper . **M. Ali Babar:** defined the scope of the project and edited the apper..

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

We thank Victor Prokhorenko, Faheem Ullah and anonymous reviewers for their valuable comments in improving the quality of the paper.

References

[1] N. Abbas, Y. Zhang, A. Taherkordi, T. Skeie, Mobile edge computing: A survey, IEEE Internet Things J. 5 (1) (2018) 450–465.
[2] A. Abdul Majeed, P. Kilpatrick, I. Spence, B. Varghese, Performance estimation of container-based cloud-to-fog offloading, in: Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing Companion, UCC 19 Companion, Association for Computing Machinery, New York, NY, USA, 2019, pp. 151–156.
[3] K.H. Abdulkareem, M.A. Mohammed, S.S. Gunasekaran, M.N. Al-Mhiqani, A.A. Mutlag, S.A. Mostafa, N.S. Ali, D.A. Ibrahim, A review of fog computing and machine learning: Concepts, applications, challenges, and open issues, IEEE Access 7 (2019) 153123–153140.
[4] S. Agarwal, S. Yadav, A.K. Yadav, An efficient architecture and algorithm for resource provisioning in fog computing, Int. J. Inf. Eng. Electron. Bus. 8 (1) (2016) 48.
[5] R.W. Ahmad, A. Gani, S.H.A. Hamid, M. Shiraz, A. Yousafzai, F. Xia, A survey on virtual machine migration and server consolidation frameworks for cloud data centers, J. Netw. Comput. Appl. 52 (C) (2015) 11–25.
[6] A. Ahmed, E. Ahmed, A survey on mobile edge computing, in: 2016 10th International Conference on Intelligent Systems and Control, ISCO, 2016, pp. 1–8.
[7] F. Ait Salaht, F. Desprez, A. Lebre, An Overview of Service Placement Problem in Fog and Edge Computing, Research Report RR-9295, Univ Lyon, EnsL, UCBL, CNRS, Inria, LIP, LYON, France, 2019, p. 43, URL https://hal.inria.fr/hal-02313711.
[8] F. Ait Salaht, F. Desprez, A. Lebre, C. Prud'homme, M. Abderrahim, Service placement in fog computing using constraint programming, in: 2019 IEEE International Conference on Services Computing, SCC, 2019, pp. 19–27.
[9] S. Akoush, R. Sohan, A. Rice, A.W. Moore, A. Hopper, Predicting the performance of virtual machine migration, in: 2010 IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, 2010, pp. 37–46.
[10] M. Al-Fares, A. Loukissas, A. Vahdat, A scalable, commodity data center network architecture, SIGCOMM Comput. Commun. Rev. 38 (4) (2008) 63–74.
[11] M. Alam, J. Rufino, J. Ferreira, S.H. Ahmed, N. Shah, Y. Chen, Orchestration of microservices for IoT using docker and edge computing, IEEE Commun. Mag. 56 (9) (2018) 118–123.

[12] I. Alam, K. Sharif, F. Li, Z. Latif, M.M. Karim, S. Biswas, B. Nour, Y. Wang, A survey of network virtualization techniques for Internet of Things using SDN and NFV, ACM Comput. Surv. 53 (2) (2020).
[13] I.A. Alimi, A.L. Teixeira, P.P. Monteiro, Toward an efficient C-RAN optical fronthaul for the future networks: A tutorial on technologies, requirements, challenges, and solutions, IEEE Commun. Surv. Tutor. 20 (1) (2018) 708–769.
[14] I. Althamary, C. Huang, P. Lin, S. Yang, C. Cheng, Popularity-based cache placement for fog networks, in: 2018 14th International Wireless Communications Mobile Computing Conference, IWCMC, 2018, pp. 800–804.
[15] G. Amarasinghe, M.D. de Assunção, A. Harwood, S. Karunasekera, A data stream processing optimisation framework for edge computing applications, in: 2018 IEEE 21st International Symposium on Real-Time Distributed Computing, ISORC, 2018, pp. 91–98.
[16] Amazon CloudFront, https://aws.amazon.com/cloudfront.
[17] F. Arena, G. Pau, An overview of vehicular communications, Future Internet 11 (2019) 27.
[18] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, M. Zaharia, A view of cloud computing, Commun. ACM 53 (4) (2010) 50–58.
[19] R.G. Aryal, J. Altmann, Dynamic application deployment in federations of clouds and edge resources using a multiobjective optimization AI algorithm, in: 2018 Third International Conference on Fog and Mobile Edge Computing, FMEC, 2018, pp. 147–154.
[20] M. Ashouri, F. Lorig, P. Davidsson, R. Spalazzese, Edge computing simulators for IoT system design: An analysis of qualities and metrics, Future Internet 11 (11) (2019) 235.
[21] M.S. Aslanpour, S.S. Gill, A. Toosi, Performance evaluation metrics for cloud, fog and edge computing: A review, taxonomy, benchmarks and standards for future research, Internet of Things 12 (2020) 100273, http://dx.doi.org/10.1016/j.iot.2020.100273.
[22] I. Azimi, A. Anzanpour, A.M. Rahmani, T. Pahikkala, M. Levorato, P. Liljeberg, N. Dutt, HiCH: Hierarchical Fog-assisted computing architecture for healthcare IoT, ACM Trans. Embed. Comput. Syst. 16 (5s) (2017).
[23] Azure Conetent Delivery Network (CDN), https://azure.microsoft.com/en-au/services/cdn/.
[24] N.G. Bachiega, P.S.L. Souza, S.M. Bruschi, S. d. R. S. de Souza, Container-based performance evaluation: A survey and challenges, in: 2018 IEEE International Conference on Cloud Engineering, IC2E, 2018, pp. 398–403.
[25] T. Bahreini, D. Grosu, Efficient placement of multi-component applications in edge computing systems, in: Proceedings of the Second ACM/IEEE Symposium on Edge Computing, SEC '17, Association for Computing Machinery, New York, NY, USA, 2017.
[26] D. Balouek, A.C. Amarie, G. Charrier, F. Desprez, E. Jeannot, E. Jeanvoine, A. Lèbre, D. Margery, N. Niclausse, L. Nussbaum, O. Richard, C. Perez, F. Quesnel, C. Rohr, L. Sarzyniec, Adding virtualization capabilities to the grid'5000 testbed, in: I.I. Ivanov, M. van Sinderen, F. Leymann, T. Shan (Eds.), Cloud Computing and Services Science, Springer International Publishing, Cham, 2013, pp. 3–20.
[27] M. Barcelo, A. Correa, J. Llorca, A.M. Tulino, J.L. Vicario, A. Morell, IoT-cloud service optimization in next generation smart environments, IEEE J. Sel. Areas Commun. 34 (12) (2016) 4077–4090.
[28] S. Barua, R. Braun, A novel approach of mobility management for the D2D communications in 5G mobile cellular network system, in: 2016 18th Asia-Pacific Network Operations and Management Symposium, APNOMS, 2016, pp. 1–4.
[29] M.T. Beck, M. Werner, S. Feld, T. Schimper, Mobile edge computing: A taxonomy in:, The Sixth International Conference on Advances in Future Internet, 2014.
[30] Z. Becvar, J. Plachy, P. Mach, Path selection using handover in mobile networks with cloud-enabled small cells, in: 2014 IEEE 25th Annual International Symposium on Personal, Indoor, and Mobile Radio Communication, PIMRC, 2014, pp. 1480–1485.
[31] R. Behravesh, E. Coronado, D. Harutyunyan, R. Riggio, Joint user association and VNF placement for latency sensitive applications in 5G networks, in: 2019 IEEE 8th International Conference on Cloud Networking, CloudNet, 2019, pp. 1–7.
[32] P. Bellavista, A. Zanni, Feasibility of fog computing deployment based on docker containerization over raspberrypi, in: Proceedings of the 18th International Conference on Distributed Computing and Networking, ICDCN 17, Association for Computing Machinery, New York, NY, USA, 2017.
[33] A. Beloglazov, R. Buyya, Energy efficient allocation of virtual machines in cloud data centers, in: Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, CCGRID '10, IEEE Computer Society, Washington, DC, USA, 2010, pp. 577–578.
[34] A.R. Benamer, H. Teyeb, N. Ben-Hadj-Alouane, Latency-aware placement heuristic in fog computing environment, in: H. Panetto, C. Debruyne, H.A. Proper, C.A. Ardagna, D. Roman, R. Meersman (Eds.), On the Move To Meaningful Internet Systems. OTM 2018 Conferences, Springer International Publishing, Cham, 2018, pp. 241–257.
[35] D. Bhamare, R. Jain, M. Samaka, A. Erbad, A survey on service function chaining, J. Netw. Comput. Appl. 75 (2016) 138–155.
[36] D. Bhamare, M. Samaka, A. Erbad, R. Jain, L. Gupta, H.A. Chan, Optimal virtual network function placement in multi-cloud service function chaining architecture, Comput. Commun. 102 (2017) 1–16.
[37] L.F. Bittencourt, M.M. Lopes, I. Petri, O.F. Rana, Towards virtual machine migration in fog computing, in: 2015 10th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, 3PGCIC, 2015, pp. 1–8.
[38] C. Bockelmann, N.K. Pratas, G. Wunder, S. Saur, M. Navarro, D. Gregoratti, G. Vivier, E. De Carvalho, Y. Ji, Stefanovi, P. Popovski, Q. Wang, M. Schellmann, E. Kosmatos, P. Demestichas, M. Raceala-Motoc, P. Jung, S. Stanczak, A. Dekorsy, Towards massive connectivity support for scalable mMTC communications in 5g networks, IEEE Access 6 (2018) 28969–28992.
[39] M.S. Bonfim, K.L. Dias, S.F.L. Fernandes, Integrated NFV/SDN architectures: A systematic literature review, ACM Comput. Surv. 51 (6) (2019) 114:1–114:39.
[40] F. Bonomi, R. Milito, J. Zhu, S. Addepalli, Fog computing and its role in the Internet of Things, in: Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing, MCC '12, ACM, New York, NY, USA, 2012, pp. 13–16.
[41] A. Brogi, S. Forti, F. Paganelli, Probabilistic QoS-aware placement of VNF chains at the edge, 2019, arXiv:abs/1906.00197.
[42] R. Buyya, C.S. Yeo, S. Venugopal, J. Broberg, I. Brandic, Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility, Future Gener. Comput. Syst. 25 (6) (2009) 599–616.
[43] R.N. Calheiros, R. Ranjan, A. Beloglazov, C.A.F. De Rose, R. Buyya, Cloudsim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms, Softw. Pract. Exper. 41 (1) (2011) 23–50.
[44] M. Capra, R. Peloso, G. Masera, M. Ruo Roch, M. Martina, Edge computing: A survey on the hardware requirements in the Internet of Things world, Future Internet 11 (2019) 100.
[45] V. Cardellini, V. Grassi, F. Lo Presti, M. Nardelli, Optimal operator placement for distributed stream processing applications, in: Proceedings of the 10th ACM International Conference on Distributed and Event-Based Systems, DEBS '16, Association for Computing Machinery, New York, NY, USA, 2016, pp. 69–80.
[46] A. Celesti, D. Mulfari, M. Fazio, M. Villari, A. Puliafito, Exploring container virtualization in IoT clouds, in: 2016 IEEE International Conference on Smart Computing, SMARTCOMP, 2016, pp. 1–6.
[47] H. Chang, A. Hari, S. Mukherjee, T.V. Lakshman, Bringing the cloud to the edge, in: 2014 IEEE Conference on Computer Communications Workshops, INFOCOM WKSHPS, 2014, pp. 346–351.
[48] L. Chaufournier, P. Sharma, F. Le, E. Nahum, P. Shenoy, D. Towsley, Fast transparent virtual machine migration in distributed edge clouds, in: Proceedings of the Second ACM/IEEE Symposium on Edge Computing, SEC 17, Association for Computing Machinery, New York, NY, USA, 2017.
[49] A. Checko, H.L. Christiansen, Y. Yan, L. Scolari, G. Kardaras, M.S. Berger, L. Dittmann, Cloud RAN for mobile networks technology overview, IEEE Commun. Surv. Tutor. 17 (1) (2015) 405–426.
[50] D. Chemodanov, P. Calyam, F. Esposito, A near optimal reliable composition approach for geo-distributed latency-sensitive service chains, in: IEEE INFOCOM 2019 - IEEE Conference on Computer Communications, 2019, pp. 1792–1800.
[51] Z. Chen, S. Zhang, C. Wang, Z. Qian, M. Xiao, J. Wu, I. Jawhar, A novel algorithm for NFV chain placement in edge computing environments, in: 2018 IEEE Global Communications Conference, GLOBECOM, 2018, pp. 1–6.
[52] C. Clark, K. Fraser, S. Hand, J.G. Hansen, E. Jul, C. Limpach, I. Pratt, A. Warfield, Live migration of virtual machines, in: Proceedings of the 2Nd Conference on Symposium on Networked Systems Design & Implementation - Volume 2, NSDI'05, USENIX Association, Berkeley, CA, USA, 2005, pp. 273–286.
[53] P. Cong, J. Zhou, L. Li, K. Cao, T. Wei, K. Li, A survey of hierarchical energy optimization for mobile edge computing: A perspective from end devices to the cloud, ACM Comput. Surv. 53 (2) (2020).
[54] J. Cosmas, B. Meunier, K. Ali, N. Jawad, M. Salih, H.-Y. Meng, J. Royo, P. Fernandez, Z. Hadad, H. Gokmen, et al., A scaleable and license free 5g internet of radio light architecture for services in train stations, in: European Wireless 2018; 24th European Wireless Conference, VDE, 2018, pp. 1–6.
[55] R. Cziva, C. Anagnostopoulos, D.P. Pezaros, Dynamic, latency-optimal vNF placement at the network edge, in: IEEE INFOCOM 2018 - IEEE Conference on Computer Communications, 2018, pp. 693–701.
[56] R. Cziva, D.P. Pezaros, Container network functions: Bringing NFV to the network edge, IEEE Commun. Mag. 55 (6) (2017) 24–31.
[57] A. Damianou, C.M. Angelopoulos, V. Katos, An architecture for blockchain over edge-enabled IoT for smart circular cities, in: 2019 15th International Conference on Distributed Computing in Sensor Systems, DCOSS, 2019, pp. 465–472.

[58] S.K. Datta, C. Bonnet, N. Nikaein, An IoT gateway centric architecture to provide novel M2M services, in: 2014 IEEE World Forum on Internet of Things, WF-IoT 2014, 2014, pp. 514–519.
[59] V. De Maio, I. Brandic, First hop mobile offloading of DAG computations, in: Proceedings of the 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, IEEE Press, 2018, pp. 83–92.
[60] P. Desai, A. Sheth, P. Anantharam, Semantic gateway as a service architecture for IoT interoperability, in: 2015 IEEE International Conference on Mobile Services, 2015, pp. 313–319.
[61] J. Dizdarević, F. Carpio, A. Jukan, X. Masip-Bruin, A survey of communication protocols for Internet of Things and related challenges of fog and cloud computing integration, ACM Comput. Surv. 51 (6) (2019).
[62] K. Dolui, S.K. Datta, Comparison of edge computing implementations: Fog computing, cloudlet and mobile edge computing, in: 2017 Global Internet of Things Summit, GIoTS, 2017, pp. 1–6.
[63] K. Dolui, C. Kiraly, Towards multi-container deployment on IoT gateways, in: 2018 IEEE Global Communications Conference, GLOBECOM, IEEE, 2018, pp. 1–7.
[64] B. Donassolo, I. Fajjari, A. Legrand, P. Mertikopoulos, Fog based framework for IoT service provisioning, in: 2019 16th IEEE Annual Consumer Communications Networking Conference, CCNC, 2019, pp. 1–6.
[65] M. Dorigo, T. Stützle, The ant colony optimization metaheuristic: Algorithms, applications, and advances, in: F. Glover, G.A. Kochenberger (Eds.), Handbook of Metaheuristics, Springer US, Boston, MA, 2003, pp. 250–285.
[66] S. Drxler, H. Karl, Z. Mann, Joint optimization of scaling and placement of virtual network services, in: 2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, CCGRID, 2017, pp. 365–370.
[67] Q. Duan, Cloud service performance evaluation: status, challenges, and opportunities a survey from the system modeling perspective, Digit. Commun. Netw. 3 (2) (2017) 101–111.
[68] A. Elhabbash, F. Samreen, J. Hadley, Y. Elkhatib, Cloud brokerage: A systematic survey, ACM Comput. Surv. 51 (6) (2019).
[69] R. Enns, M. Bjorklund, J. Schoenwaelder, A. Bierman, Network configuration protocol (NETCONF), RFC 6241, 2011, https://www.rfc-editor.org/rfc/rfc6241.txt.
[70] V. Eramo, E. Miucci, M. Ammar, F.G. Lavacca, An approach for service function chain routing and virtual function network instance migration in network function virtualization architectures, IEEE/ACM Trans. Netw. 25 (4) (2017) 2008–2025.
[71] W. Felter, A. Ferreira, R. Rajamony, J. Rubio, An updated performance comparison of virtual machines and Linux containers, in: 2015 IEEE International Symposium on Performance Analysis of Systems and Software, ISPASS, 2015, pp. 171–172.
[72] N. Fernando, S.W. Loke, W. Rahayu, Mobile cloud computing: A survey, Future Gener. Comput. Syst. 29 (1) (2013) 84–106, Including Special section: AIRCC-NetCoM 2009 and Special section: Clouds and Service-Oriented Architectures.
[73] S. Filiposka, A. Mishev, K. Gilly, Community-based allocation and migration strategies for fog computing, in: 2018 IEEE Wireless Communications and Networking Conference, WCNC, 2018, pp. 1–6.
[74] M. Forsman, A. Glad, L. Lundberg, D. Ilie, Algorithms for automated live migration of virtual machines, J. Syst. Softw. 101 (2015) 110–126.
[75] I. Foster, Y. Zhao, I. Raicu, S. Lu, Cloud computing and grid computing 360-degree compared, in: 2008 Grid Computing Environments Workshop, 2008, pp. 1–10.
[76] T. Frassetto, P. Jauernig, C. Liebchen, A.-R. Sadeghi, IMIX: In-process memory isolation extension, in: 27th USENIX Security Symposium, 2018, pp. 83–97.
[77] R. Gargees, B. Morago, R. Pelapur, D. Chemodanov, P. Calyam, Z. Oraibi, Y. Duan, G. Seetharaman, K. Palaniappan, Incident-supporting visual cloud computing utilizing software-defined networking, IEEE Trans. Circuits Syst. Video Technol. 27 (1) (2017) 182–197.
[78] C.S. Gates, N. Li, J. Chen, R.W. Proctor, Codeshield: towards personalized application whitelisting, in: R.H. Zakon (Ed.), 28th Annual Computer Security Applications Conference, ACSAC 2012, Orlando, FL, USA, 3–7 December 2012, ACM, 2012, pp. 279–288.
[79] J. Gedeon, M. Stein, L. Wang, M. Muehlhaeuser, On scalable in-network operator placement for edge computing, 2018 27th International Conference on Computer Communication and Networks, ICCCN, 2018, pp. 1–9.
[80] M. Ghaznavi, N. Shahriar, S. Kamali, R. Ahmed, R. Boutaba, Distributed service function chaining, IEEE J. Sel. Areas Commun. 35 (11) (2017) 2479–2489.
[81] M. Ghobaei-Arani, A. Souri, A.A. Rahmanian, Resource management approaches in fog computing: a comprehensive review, J. Grid Comput. 18 (2020) 1–42.
[82] A.M. Ghosh, K. Grolinger, Deep learning: Edge-cloud data analytics for IoT, in: 2019 IEEE Canadian Conference of Electrical and Computer Engineering, CCECE, 2019, pp. 1–7.
[83] R. Ghosh2018, S.P.R. Komma, Y. Simmhan, Adaptive energy-aware scheduling of dynamic event analytics across edge and cloud resources, in: Proceedings of the 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, CCGrid '18, IEEE Press, 2018, pp. 72–82.
[84] S.S. Gill, S. Tuli, M. Xu, I. Singh, K.V. Singh, D. Lindsay, S. Tuli, D. Smirnova, M. Singh, U. Jain, H. Pervaiz, B. Sehgal, S.S. Kaila, S. Misra, M.S. Aslanpour, H. Mehta, V. Stankovski, P. Garraghan, Transformative effects of IoT, blockchain and artificial intelligence on cloud computing: Evolution, vision, trends and open challenges, Internet of Things 8 (2019) 100118.
[85] R.P. Goldberg, Survey of virtual machine research, Computer 7 (9) (1974) 34–45.
[86] D. Gonalves, K. Velasquez, M. Curado, L. Bittencourt, E. Madeira, Proactive virtual machine migration in fog environments, in: 2018 IEEE Symposium on Computers and Communications, ISCC, 2018 pp. 00742–00745.
[87] D. Gonalves, K. Velasquez, M. Curado, L. Bittencourt, E. Madeira, Proactive virtual machine migration in fog environments, in: 2018 IEEE Symposium on Computers and Communications, ISCC, 2018, pp. 00742–00745.
[88] Google Content Delivery Network (CDN) https://cloud.google.com/cdn/.
[89] S. Guoqiang, C. Yanming, Z. Chao, Z. Yanxu, Design and implementation of a smart IoT gateway, in: 2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing, 2013, pp. 720–723.
[90] H. Gupta, A. Dastjerdi, S. Ghosh, R. Buyya, iFogSim: A Toolkit for modeling and simulation of resource management techniques in Internet of Things, edge and fog computing environments, Softw. - Pract. Exp. (2016).
[91] K. Ha, Y. Abe, T. Eiszler, Z. Chen, W. Hu, B. Amos, R. Upadhyaya, P. Pillai, M. Satyanarayanan, You can teach elephants to dance: Agile VM Handoff for edge computing, in: Proceedings of the Second ACM/IEEE Symposium on Edge Computing, SEC '17, ACM, New York, NY, USA, 2017, pp. 12:1–12:14.
[92] K. Ha, P. Pillai, W. Richter, Y. Abe, M. Satyanarayanan, Just-in-time provisioning for cyber foraging, in: Proceeding of the 11th Annual International Conference on Mobile Systems, Applications, and Services, MobiSys '13, ACM, New York, NY, USA, 2013, pp. 153–166.
[93] M.A. Habibi, M. Nasimi, B. Han, H.D. Schotten, A comprehensive survey of RAN architectures toward 5g mobile communication system, IEEE Access 7 (2019) 70371–70421.
[94] K.J. Han, B.-Y. Choi, S. Song, High Performance Cloud Auditing and Applications, first ed., Springer Publishing Company, Incorporated, 2016.
[95] E. Harjula, P. Karhula, J. Islam, T. Leppänen, A. Manzoor, M. Liyanage, J. Chauhan, T. Kumar, I. Ahmad, M. Ylianttila, Decentralized Iot edge nanoservice architecture for future gadget-free computing, IEEE Access 7 (2019) 119856–119872.
[96] M. Hartmann, U.S. Hashmi, A. Imran, Edge computing in smart health care systems: Review, challenges, and research directions, Trans. Emerg. Telecommun. Technol. n/a (n/a) (2019) e3710.
[97] M. Hassan, M. Xiao, Q. Wei, S. Chen, Help your mobile applications with fog computing, in: 2015 12th Annual IEEE International Conference on Sensing, Communication, and Networking - Workshops, SECON Workshops, 2015, pp. 1–6.
[98] J.H. Holland, Bibliography, in: Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications To Biology, Control, and Artificial Intelligence, 1992, pp. 203–205.
[99] H. Hong, P. Tsai, C. Hsu, Dynamic module deployment in a fog computing platform, in: 2016 18th Asia-Pacific Network Operations and Management Symposium, APNOMS, 2016, pp. 1–6.
[100] C.-H. Hong, B. Varghese, Resource management in fog/edge computing: A survey on architectures, infrastructure, and algorithms, ACM Comput. Surv. 52 (5) (2019).
[101] J. Hubaux, T. Gross, J. Le Boudec, M. Vetterli, Toward self-organized mobile ad hoc networks: the terminodes project, IEEE Commun. Mag. 39 (1) (2001) 118–124.
[102] Intel Software Guard Extensions, https://software.intel.com/en-us/sgx.
[103] K. Jain, S. Mohapatra, Taxonomy of edge computing: Challenges, opportunities, and data reduction methods: From hype to reality, 2019, pp. 51–69.
[104] S.Y. Jang, Y. Lee, B. Shin, D. Lee, Application-aware IoT camera virtualization for video analytics edge computing, in: 2018 IEEE/ACM Symposium on Edge Computing, SEC, 2018, pp. 132–144.
[105] S.J. Johnston, P.J. Basford, C.S. Perkins, H. Herry, F.P. Tso, D. Pezaros, R.D. Mullins, E. Yoneki, S.J. Cox, J. Singer, Commodity single board computer clusters and their applications, Future Gener. Comput. Syst. 89 (2018) 201–212.
[106] S.J. Johnston, P.J. Basford, C.S. Perkins, H. Herry, F.P. Tso, D. Pezaros, R.D. Mullins, E. Yoneki, S.J. Cox, J. Singer, Commodity single board computer clusters and their applications, Future Gener. Comput. Syst. 89 (2018) 201–212.

[107] S.R.U. Kakakhel, L. Mukkala, T. Westerlund, J. Plosila, Virtualization at the network edge: A technology perspective, in: 2018 Third International Conference on Fog and Mobile Edge Computing, FMEC, 2018, pp. 87–92.
[108] S.R.U. Kakakhel, L. Mukkala, T. Westerlund, J. Plosila, Virtualization at the network edge: A technology perspective, in: 2018 Third International Conference on Fog and Mobile Edge Computing, FMEC, 2018, pp. 87–92.
[109] P. Karhula, J. Mäkelä, H. Rivas, M. Valta, Internet of Things connectivity with gateway functionality virtualization, in: 2017 Global Internet of Things Summit (GIoTS), 2017, pp. 1–6.
[110] K. Kaur, T. Dhand, N. Kumar, S. Zeadally, Container-as-a-service at the edge: Trade-off between energy efficiency and service availability at fog nano data centers, IEEE Wirel. Commun. 24 (2017) 48–56.
[111] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of ICNN'95 - International Conference on Neural Networks, vol. 4, 1995, pp. 1942–1948.
[112] I. Khalil, A. Khreishah, M. Azeem, Cloud computing security: A survey, Computers 3 (1) (2014) 1–35.
[113] W.Z. Khan, E. Ahmed, S. Hakak, I. Yaqoob, A. Ahmed, Edge computing: A survey, Future Gener. Comput. Syst. 97 (2019) 219–235.
[114] S. Khan, S. Parkinson, Y. Qin, Fog computing security: A review of current applications and security solutions, J. Cloud Comput. 6 (1) (2017).
[115] K. Khanda, D. Salikhov, K. Gusmanov, M. Mazzara, N. Mavridis, Microservice-based IoT for smart buildings, in: 2017 31st International Conference on Advanced Information Networking and Applications Workshops, WAINA, 2017, pp. 302–308.
[116] V. Kochar, A. Sarkar, Real time resource allocation on a dynamic two level symbiotic fog architecture, in: 2016 Sixth International Symposium on Embedded Computing and System Design, ISED, 2016, pp. 49–55.
[117] D. Kreutz, F.M.V. Ramos, P.E. Verssimo, C.E. Rothenberg, S. Azodolmolky, S. Uhlig, Software-defined networking: A comprehensive survey, Proc. IEEE 103 (1) (2015) 14–76.
[118] A. Krylovskiy, Internet of Things gateways meet linux containers: Performance evaluation and discussion, in: 2015 IEEE 2nd World Forum on Internet of Things, WF-IoT, 2015, pp. 222–227.
[119] Kubernetes for container orchestration, https://kubernetes.io/.
[120] T. Kumar, E. Harjula, M. Ejaz, A. Manzoor, P. Porambage, I. Ahmad, M. Liyanage, A. Braeken, M. Ylianttila, Blockedge: Blockchain-edge framework for industrial IoT networks, IEEE Access 8 (2020) 154166–154185.
[121] KVM, https://www.linux-kvm.org/page/Main_Page.
[122] S. Lal, T. Taleb, A. Dutta, NFV: Security threats and best practices, IEEE Commun. Mag. 55 (8) (2017) 211–217.
[123] K. Lee, H. Kim, B. Kim, C. Yoo, Analysis on network performance of container virtualization on IoT devices, in: 2017 International Conference on Information and Communication Technology Convergence, ICTC, 2017, pp. 35–37.
[124] A. Leivadeas, G. Kesidis, M. Ibnkahla, I. Lambadaris, VNF placement optimization at the edge and cloud, Future Internet 11 (3) (2019).
[125] D. Li, P. Hong, K. Xue, J. Pei, Virtual network function placement and resource optimization in NFV and edge computing enabled networks, Comput. Netw. 152 (2019) 12–24.
[126] Y. Li, T. Jiang, K. Luo, S. Mao, Green heterogeneous cloud radio access networks: Potential techniques, performance trade-offs, and challenges, IEEE Commun. Mag. 55 (11) (2017) 33–39.
[127] J. Li, J. Jin, D. Yuan, H. Zhang, Virtual fog: A virtualization enabled fog computing framework for Internet of Things, IEEE Internet Things J. 5 (1) (2018) 121–131.
[128] F. Liu, G. Tang, Y. Li, Z. Cai, X. Zhang, T. Zhou, A survey on edge computing systems and tools, Proc. IEEE 107 (8) (2019) 1537–1562.
[129] M.M. Lopes, W.A. Higashino, M.A. Capretz, L.F. Bittencourt, MyiFogSim: A simulator for virtual machine migration in fog computing, in: Companion Proceedings of The10th International Conference on Utility and Cloud Computing, UCC 17 Companion, Association for Computing Machinery, New York, NY, USA, 2017, pp. 47–52.
[130] C. Luo, L. Xu, D. Li, W. Wu, Edge computing integrated with blockchain technologies, in: D.-Z. Du, J. Wang (Eds.), Complexity and Approximation: In Memory of Ker-I Ko, Springer International Publishing, Cham, 2020, pp. 268–288.
[131] X. Lyu, H. Tian, W. Ni, Y. Zhang, P. Zhang, R. Liu, Energy-efficient admission of delay-sensitive tasks for mobile edge computing, IEEE Trans. Commun. PP (2018).
[132] L. Ma, S. Yi, Q. Li, Efficient service handoff across edge servers via docker container migration, in: Proceedings of the Second ACM/IEEE Symposium on Edge Computing, in: SEC 17, Association for Computing Machinery, New York, NY, USA, 2017.
[133] P. Mach, Z. Becvar, Mobile edge computing: A survey on architecture and computation offloading, IEEE Commun. Surv. Tutor. 19 (3) (2017) 1628–1656.
[134] A. Machen, S. Wang, K.K. Leung, B.J. Ko, T. Salonidis, Live service migration in mobile edge clouds, IEEE Wirel. Commun. 25 (1) (2018) 140–147.
[135] A. Madhavapeddy, R. Mortier, C. Rotsos, D. Scott, B. Singh, T. Gazagnaire, S. Smith, S. Hand, J. Crowcroft, Unikernels: Library operating systems for the cloud, SIGPLAN Not. 48 (4) (2013) 461–472.
[136] A. Madhavapeddy, R. Mortier, C. Rotsos, D. Scott, B. Singh, T. Gazagnaire, S. Smith, S. Hand, J. Crowcroft, Unikernels: Library operating systems for the cloud, SIGARCH Comput. Archit. News 41 (1) (2013) 461–472.
[137] R. Mahmud, K. Ramamohanarao, R. Buyya, Latency-aware application module management for fog computing environments, ACM Trans. Internet Technol. 19 (1) (2018).
[138] Cloudband application manager, https://www.nokia.com/networks/products/cloudband-application-manager/.
[139] Y. Mansouri, R. Buyya, Dynamic replication and migration of data objects with hot-spot and cold-spot statuses across storage data centers, J. Parallel Distrib. Comput. 126 (2019) 121–133.
[140] Y. Mansouri, A.N. Toosi, R. Buyya, Brokering algorithms for optimizing the availability and cost of cloud storage services, in: Proceedings of the 2013 IEEE International Conference on Cloud Computing Technology and Science - Volume 01, CLOUDCOM '13, IEEE Computer Society, Washington, DC, USA, 2013, pp. 581–589.
[141] Y. Mansouri, A.N. Toosi, R. Buyya, Data storage management in cloud environments: Taxonomy, survey, and future directions, ACM Comput. Surv. 50 (6) (2017) 91:1–91:51.
[142] Y. Mansouri, A.N. Toosi, R. Buyya, Cost optimization for dynamic replication and migration of data in cloud data centers, IEEE Trans. Cloud Comput. 7 (3) (2019) 705–718.
[143] Y. Mao, C. You, J. Zhang, K. Huang, K.B. Letaief, A survey on mobile edge computing: The communication perspective, IEEE Commun. Surv. Tutor. 19 (4) (2017) 2322–2358.
[144] E. Marín-Tordera, X. Masip-Bruin, J.G. Almiñana, A. Jukan, G.-J. Ren, J. Zhu, Do we all really know what a fog node is? Current trends towards an open definition, Comput. Commun. 109 (2017) 117–130.
[145] J. Martins, M. Ahmed, C. Raiciu, V. Olteanu, M. Honda, R. Bifulco, F. Huici, Clickos and the art of network function virtualization, in: Proceedings of the 11th USENIX Conference on Networked Systems Design and Implementation, NSDI'14, USENIX Association, Berkeley, CA, USA, 2014, pp. 459–473.
[146] T. Mastelic, A. Oleksiak, H. Claussen, I. Brandic, J.-M. Pierson, A.V. Vasilakos, Cloud computing: Survey on energy efficiency, ACM Comput. Surv. 47 (2) (2014).
[147] R. Mayer, H. Gupta, E. Saurez, U. Ramachandran, FogStore: Toward a distributed data store for Fog computing, in: 2017 IEEE Fog World Congress, FWC, 2017, pp. 1–6.
[148] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, J. Turner, Openflow: Enabling innovation in campus networks, SIGCOMM Comput. Commun. Rev. 38 (2) (2008) 69–74.
[149] S. Memon, M. Maheswaran, Using machine learning for handover optimization in vehicular fog computing, in: Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing, SAC 19, Association for Computing Machinery, New York, NY, USA, 2019, pp. 182–190.
[150] R. Mijumbi, J. Serrat, J. Gorricho, S. Latre, M. Charalambides, D. Lopez, Management and orchestration challenges in network functions virtualization, IEEE Commun. Mag. 54 (1) (2016) 98–105.
[151] Q.T. Minh, D.T. Nguyen, A. Van Le, H.D. Nguyen, A. Truong, Toward service placement on Fog computing landscape, in: 2017 4th NAFOSTED Conference on Information and Computer Science, 2017, pp. 291–296.
[152] G. Miotto, M.C. Luizelli, W.L.d.C. Cordeiro, L.P. Gaspary, Adaptive placement & chaining of virtual network functions with NFV-PEAR, J. Internet Serv. Appl. 10 (1) (2019) 3.
[153] Mobile Edge Computing (MEC): Framework and Reference Architecture. ETSI GS MEC 003 v1.1.1 (Mar. 2016), http://www.etsi.org/deliver/etsi_gs/MEC/001_099/003/01.01.01_60/gs_MEC003v010101p.pd.
[154] Mobile Edge Computing (MEC): Technical Requirements. ETSI GS MEC 002 v1.1.1 (Mar. 2016), www.etsi.org/deliver/etsi_gs/MEC/001_099/002/01.01.01_60/gs_MEC002v010101p.pdf.
[155] J.C. Mogul, R.R. Kompella, Inferring the network latency requirements of cloud tenants, in: Proceedings of the 15th USENIX Conference on Hot Topics in Operating Systems, HOTOS15, USENIX Association, USA, 2015, p. 24.
[156] N. Mohan, P. Zhou, K. Govindaraj, J. Kangasharju, Managing data in computational edge clouds, in: Proceedings of the Workshop on Mobile Edge Communications, MECOMM '17, Association for Computing Machinery, New York, NY, USA, 2017, pp. 19–24.
[157] R. Morabito, Virtualization on Internet of Things edge devices with container technologies: A performance evaluation, IEEE Access 5 (2017) 8835–8850.
[158] R. Morabito, Lightweight Virtualization in Edge Computing for Internet of Things (Ph.D. thesis), 2019.
[159] R. Morabito, V. Cozzolino, A.Y. Ding, N. Beijar, J. Ott, Consolidate IoT edge computing with lightweight virtualization, IEEE Netw. 32 (1) (2018) 102–111.

[160] R. Morabito, I. Farris, A. Iera, T. Taleb, Evaluating performance of containerized IoT services for clustered devices at the network edge, IEEE Internet Things J. 4 (4) (2017) 1019–1030.
[161] R. Morabito, R. Petrolo, V. Loscr, N. Mitton, LEGIoT: A lightweight edge gateway for the Internet of Things, Future Gener. Comput. Syst. 81 (2018) 1–15.
[162] C. Mouradian, D. Naboulsi, S. Yangui, R.H. Glitho, M.J. Morrow, P.A. Polakos, A comprehensive survey on fog computing: State-of-the-art and research challenges, IEEE Commun. Surv. Tutor. 20 (1) (2018) 416–464.
[163] S. Muralidhar, W. Lloyd, S. Roy, C. Hill, E. Lin, W. Liu, S. Pan, S. Shankar, V. Sivakumar, L. Tang, S. Kumar, F4: Facebook's warm BLOB storage system, in: Proceedings of the 11th USENIX Symposium on Operating Systems Design and Implementation, OSDI 14, USENIX Association, Broomfield, CO, 2014, pp. 383–398.
[164] A.M. Mustafa, O.M. Abubakr, O. Ahmadien, A. Ahmedin, B. Mokhtar, Mobility prediction for efficient resources management in vehicular cloud computing, in: 2017 5th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering, MobileCloud, 2017, pp. 53–59.
[165] R.K. Naha, S. Garg, D. Georgakopoulos, P.P. Jayaraman, L. Gao, Y. Xiang, R. Ranjan, Fog computing: Survey of trends, architectures, requirements, and research directions, IEEE Access 6 (2018) 47980–48009.
[166] R. Naha, S. Garg, D. Georgakopoulos, P.P. Jayaraman, L. Gao, Y. Xiang, R. Ranjan, Fog computing: Survey of trends, architectures, requirements, and research directions, IEEE Access 6 (2018) 47980–48009.
[167] M. Nelson, B.-H. Lim, G. Hutchins, Fast transparent migration for virtual machines, in: Proceedings of the Annual Conference on USENIX Annual Technical Conference, ATEC '05, USENIX Association, Berkeley, CA, USA, 2005, p. 25.
[168] Network Function Virtualization (NFV), https://www.etsi.org/technologies/nfv/open-source-mano.
[169] D.T. Nguyen, C. Pham, K.K. Nguyen, M. Cheriet, Virtual network function placement in IoT network, in: 2019 15th International Wireless Communications Mobile Computing Conference, IWCMC, 2019, pp. 1166–1171.
[170] J. Ni, K. Zhang, X. Lin, X. Shen, Securing fog computing for Internet of Things applications: Challenges and solutions, IEEE Commun. Surv. Tutor. 20 (1) (2018) 601–628.
[171] V. Noronha, E. Lang, M. Riegel, T. Bauschert, Performance evaluation of container based virtualization on embedded microprocessors, in: 2018 30th International Teletraffic Congress, ITC 30, vol. 01, 2018, pp. 79–84.
[172] O. Novo, N. Beijar, M. Ocak, J. Kjällman, M. Komu, T. Kauppinen, Capillary networks - bridging the cellular and IoT worlds, in: 2015 IEEE 2nd World Forum on Internet of Things, WF-IoT, 2015, pp. 571–578.
[173] NVGRE, https://web.archive.org/web/20160527115218/http://www.definethecloud.net/nvgre.
[174] K. Ogawa, K. Kanai, K. Nakamura, H. Kanemitsu, J. Katto, H. Nakazato, IoT device virtualization for efficient resource utilization in smart city IoT platform, in: 2019 IEEE International Conference on Pervasive Computing and Communications Workshops, PerCom Workshops, 2019, pp. 419–422.
[175] P. Olivier, D. Chiba, S. Lankes, C. Min, B. Ravindran, A binary-compatible unikernel, in: Proceedings of the 15th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments, VEE 2019, Association for Computing Machinery, New York, NY, USA, 2019, pp. 59–73.
[176] Open vSwitch, http://www.openvswitch.org/features/.
[177] OPENDAYLIGHT, https://www.opendaylight.org/.
[178] OpenFV, https://www.opnfv.org/.
[179] B. Ottenwalder, B. Koldehofe, K. Rothermel, U. Ramachandran, Migcep: Operator migration for mobility driven distributed complex event processing, in: Proceedings of the 7th ACM International Conference on Distributed Event-Based Systems, DEBS '13, Association for Computing Machinery, New York, NY, USA, 2013, pp. 183–194.
[180] T. Ouyang, Z. Zhou, X. Chen, Follow me at the edge: Mobility-aware dynamic service placement for mobile edge computing, IEEE J. Sel. Areas Commun. 36 (10) (2018) 2333–2345.
[181] A. Papageorgiou, B. Cheng, E. Kovacs, Real-time data reduction at the network edge of Internet-of-Things systems, in: 2015 11th International Conference on Network and Service Management (CNSM), 2015, pp. 284–291, http://dx.doi.org/10.1109/CNSM.2015.7367373.
[182] S. Park, M.-S. Gil, H. Im, Y.-S. Moon, Measurement noise recommendation for efficient Kalman filtering over a large amount of sensor data, Sensors 19 (5) (2019) 1168.
[183] M. Peng, Y. Li, J. Jiang, J. Li, C. Wang, Heterogeneous cloud radio access networks: a new perspective for enhancing spectral and energy efficiencies, IEEE Wirel. Commun. 21 (6) (2014) 126–135.
[184] C. Perera, Y. Qin, J.C. Estrella, S. Reiff-Marganiec, A.V. Vasilakos, Fog computing for sustainable smart cities: A survey, ACM Comput. Surv. 50 (3) (2017).
[185] G. Peskir, A. Shiryaev, Optimal stopping and free-boundary problems, Lectures in Mathematics. ETH Zürich, Birkhäuser Basel, 2006.
[186] R. Petrolo, R. Morabito, V. Loscrí, N. Mitton, The design of the gateway for the Cloud of Things, Ann. Telecommun. 72 (2017) 31–40.
[187] J. Plachy, Z. Becvar, P. Mach, Path selection enabling user mobility and efficient distribution of data for computation at the edge of mobile network, Comput. Netw. 108 (2016) 357–370.
[188] J.S. Preden, K. Tammeme, A. Jantsch, M. Leier, A. Riid, E. Calis, The benefits of self-awareness and attention in fog and mist computing, Computer 48 (7) (2015) 37–45.
[189] C. Puliafito, E. Mingozzi, F. Longo, A. Puliafito, O. Rana, Fog computing for the Internet of Things: A survey, ACM Trans. Internet Technol. 19 (2) (2019) 18:1–18:41.
[190] C. Puliafito, C. Vallati, E. Mingozzi, G. Merlino, F. Longo, A. Puliafito, Container migration in the fog: A performance evaluation, Sensors 19 (2019) 1488.
[191] Y. Qiu, C. Lung, S. Ajila, P. Srivastava, LXC container migration in cloudlets under multipath TCP, in: 2017 IEEE 41st Annual Computer Software and Applications Conference, COMPSAC, vol. 2, 2017, pp. 31–36.
[192] Raspberri Pi4, in: https://www.raspberrypi.org.
[193] R. Ré, R.M. Meloca, D.N. Roma, M.A. da Cruz Ismael, G.C. Silva, An empirical study for evaluating the performance of multi-cloud APIs, Future Gener. Comput. Syst. 79 (2018) 726–738.
[194] RedHat OpenShift, https://www.openshift.com/.
[195] H. Ren, Z. Xu, W. Liang, Q. Xia, P. Zhou, O.F. Rana, A. Galis, G. Wu, Efficient algorithms for delay-aware NFV-enabled multicasting in mobile edge clouds with resource sharing, IEEE Trans. Parallel Distrib. Syst. 31 (9) (2020) 2050–2066.
[196] J. Ren, D. Zhang, S. He, Y. Zhang, T. Li, A survey on end-edge-cloud orchestrated network computing paradigms: Transparent computing, mobile edge computing, fog computing, and cloudlet, ACM Comput. Surv. 52 (6) (2019).
[197] L. Rizzo, M. Carbone, G. Catalli, Transparent acceleration of software packet forwarding using netmap, in: 2012 Proceedings IEEE INFOCOM, 2012, pp. 2471–2479.
[198] D. Roca, J.V. Quiroga, M. Valero, M. Nemirovsky, Fog function virtualization: A flexible solution for IoT applications, in: 2017 Second International Conference on Fog and Mobile Edge Computing, FMEC, 2017, pp. 74–80.
[199] R. Roman, J. Lopez, M. Mambo, Mobile edge computing, Fog et al.: A survey and analysis of security threats and challenges, Future Gener. Comput. Syst. 78 (2018) 680–698.
[200] F.A. Salaht, F. Desprez, A. Lebre, An overview of service placement problem in fog and edge computing, ACM Comput. Surv. 53 (3) (2020).
[201] M.A. Salahuddin, J. Sahoo, R. Glitho, H. Elbiaze, W. Ajib, A survey on content placement algorithms for cloud-based content delivery networks, IEEE Access 6 (2018) 91–114.
[202] M. Samaniego, R. Deters, Management and Internet of Things, Procedia Comput. Sci. 94 (2016) 137–143, The 11th International Conference on Future Networks and Communications (FNC 2016) / The 13th International Conference on Mobile Systems and Pervasive Computing (MobiSPC 2016) / Affiliated Workshops.
[203] J. Santa, J. Ortiz, P.J. Fernandez, M. Luis, C. Gomes, J. Oliveira, D. Gomes, R. Sanchez-Iborra, S. Sargento, A.F. Skarmeta, MIGRATE: Mobile device virtualisation through state transfer, IEEE Access 8 (2020) 25848–25862.
[204] M. Satyanarayanan, P. Bahl, R. Caceres, N. Davies, The case for VM-based cloudlets in mobile computing, IEEE Pervasive Comput. 8 (4) (2009) 14–23.
[205] E. Saurez, K. Hong, D. Lillethun, U. Ramachandran, B. Ottenwälder, Incremental deployment and migration of geo-distributed situation awareness applications in the fog, in: Proceedings of the 10th ACM International Conference on Distributed and Event-Based Systems, DEBS 16, Association for Computing Machinery, New York, NY, USA, 2016, pp. 258–269.
[206] SDN deployment in purely hardware network devices, https://www.google.com/search?q=Cisco+switches+virtualization&source=univ&tbm=shop&tbo=u&sa=X&ved=0ahUKEwjlIOTj2Z_kAhXYILcAHRAICesQsxgIMA#spd=7573947727014188636.
[207] J. Serra, L. Sanabria-Russo, D. Pubill, C. Verikoukis, Scalable and flexible IoT data analytics: when machine learning meets SDN and virtualization, in: 2018 IEEE 23rd International Workshop on Computer Aided Modeling and Design of Communication Links and Networks, CAMAD, 2018, pp. 1–6.
[208] P. Sharma, L. Chaufournier, P. Shenoy, Y.C. Tay, Containers and virtual machines at scale: A comparative study, in: Proceedings of the 17th International Middleware Conference, Middleware '16, ACM, New York, NY, USA, 2016, pp. 1:1–1:13.
[209] R. Shea, F. Wang, H. Wang, J. Liu, A deep investigation into network performance in virtual machine based cloud environments, in: IEEE INFOCOM 2014 - IEEE Conference on Computer Communications, 2014, pp. 1285–1293.
[210] B. Sheng, Q. Li, W. Mao, W. Jin, Outlier detection in sensor networks, MobiHoc '07, Association for Computing Machinery, New York, NY, USA, 2007, pp. 219–228.
[211] J. Shuja, A. Gani, K. Bilal, A.U.R. Khan, S.A. Madani, S.U. Khan, A.Y. Zomaya, A survey of mobile device virtualization: Taxonomy and state of the art, ACM Comput. Surv. 49 (1) (2016) 1:1–1:36.

[212] O. Skarlat, M. Nardelli, S. Schulte, M. Borkowski, P. Leitner, Optimized IoT service placement in the fog, Serv. Oriented Comput. Appl. 11 (4) (2017) 427–443.

[213] O. Skarlat, M. Nardelli, S. Schulte, S. Dustdar, Towards QoS-aware fog service placement, in: 2017 IEEE 1st International Conference on Fog and Edge Computing, ICFEC, 2017, pp. 89–96.

[214] O. Skarlat, S. Schulte, M. Borkowski, P. Leitner, Resource provisioning for IoT services in the Fog, in: 2016 IEEE 9th International Conference on Service-Oriented Computing and Applications, SOCA, 2016, pp. 32–39.

[215] J.E. Smith, Ravi Nair, The architecture of virtual machines, Computer 38 (5) (2005) 32–38.

[216] J. Soares, M. Dias, J. Carapinha, B. Parreira, S. Sargento, Cloud4NFV: A platform for virtual network functions, 2014 IEEE 3rd International Conference on Cloud Networking, CloudNet, 2014, pp. 288–293.

[217] A.H. Sodhro, Z. Luo, A.K. Sangaiah, S.W. Baik, Mobile edge computing based QoS optimization in medical healthcare applications, Int. J. Inf. Manage. 45 (2019) 308–318.

[218] J. Son, R. Buyya, Latency-aware virtualized network function provisioning for distributed edge clouds, J. Syst. Softw. 152 (2019) 24–31.

[219] V.B.C. Souza, W. Ramírez, X. Masip-Bruin, E. Marín-Tordera, G. Ren, G. Tashakor, Handling service allocation in combined Fog-cloud scenarios, in: 2016 IEEE International Conference on Communications, ICC, 2016, pp. 1–5.

[220] T. Taleb, A. Ksentini, P.A. Frangoudis, Follow-me cloud: When cloud services follow mobile users, IEEE Trans. Cloud Comput. 7 (2) (2019) 369–382.

[221] H. Tan, Z. Han, X. Li, F.C.M. Lau, Online job dispatching and scheduling in edge-clouds, in: IEEE INFOCOM 2017 - IEEE Conference on Computer Communications, 2017, pp. 1–9.

[222] M. Taneja, J. Byabazaire, N. Jalodia, A. Davy, C. Olariu, P. Malone, Machine learning based fog computing assisted data-driven approach for early lameness detection in dairy cattle, Comput. Electron. Agric. 171 (2020) 105286.

[223] M. Taneja, A. Davy, Resource aware placement of IoT application modules in Fog-Cloud Computing Paradigm, in: 2017 IFIP/IEEE Symposium on Integrated Network and Service Management, IM, 2017, pp. 1222–1228.

[224] Z. Tang, X. Zhou, F. Zhang, W. Jia, W. Zhao, Migration modeling and learning algorithms for containers in fog computing, IEEE Trans. Serv. Comput. 12 (5) (2019) 712–725.

[225] Z. Tao, Q. Xia, Z. Hao, C. Li, L. Ma, S. Yi, Q. Li, A survey of virtual machine management in edge computing, Proc. IEEE 107 (8) (2019) 1482–1499.

[226] F. Teka, C. Lung, S. Ajila, Seamless live virtual machine migration with cloudlets and multipath TCP, in: 2015 IEEE 39th Annual Computer Software and Applications Conference, vol. 2, 2015, pp. 607–616.

[227] R.I. Tinini, L.C.M. Reis, D.M. Batista, G.B. Figueiredo, M. Tornatore, B. Mukherjee, Optimal placement of virtualized BBU processing in hybrid cloud-fog RAN over TWDM-PON, in: GLOBECOM 2017 - 2017 IEEE Global Communications Conference, 2017, pp. 1–6.

[228] L. Tong, Y. Li, W. Gao, A hierarchical edge cloud architecture for mobile computing, in: IEEE INFOCOM 2016 - the 35th Annual IEEE International Conference on Computer Communications, 2016, pp. 1–9.

[229] A.N. Toosi, R.N. Calheiros, R. Buyya, Interconnected cloud computing environments: Challenges, taxonomy, and survey, ACM Comput. Surv. 47 (1) (2014) 7:1–7:47.

[230] A.N. Toosi, J. Son, Q. Chi, R. Buyya, Elasticsfc: Auto-scaling techniques for elastic service function chaining in network functions virtualization-based clouds, J. Syst. Softw. 152 (2019) 108–119.

[231] S. Tuli, N. Basumatary, S.S. Gill, M. Kahani, R.C. Arya, G.S. Wander, R. Buyya, HealthFog: An ensemble deep learning based Smart Healthcare System for Automatic Diagnosis of Heart Diseases in integrated IoT and fog computing environments, Future Gener. Comput. Syst. 104 (2020) 187–200.

[232] S. Tuli, R. Mahmud, S. Tuli, R. Buyya, FogBus: A blockchain-based lightweight framework for edge and fog computing, J. Syst. Softw. 154 (2019) 22–36.

[233] R. Uhlig, G. Neiger, D. Rodgers, A.L. Santoni, F.C.M. Martins, A.V. Anderson, S.M. Bennett, A. Kagi, F.H. Leung, L. Smith, Intel virtualization technology, Computer 38 (5) (2005) 48–56.

[234] L.M. Vaquero, L. Rodero-Merino, Finding your way in the fog: Towards a comprehensive definition of fog computing, SIGCOMM Comput. Commun. Rev. 44 (5) (2014) 27–32.

[235] B. Varghese, R. Buyya, Next generation cloud computing: New trends and research directions, Future Gener. Comput. Syst. 79 (2018) 849–861.

[236] M. Vazquez-Olguin, Y.S. Shmaliy, O. Ibarra-Manzano, J. Munoz-Minjares, C. Lastre-Dominguez, Object tracking over distributed WSNs with consensus on estimates and missing data, IEEE Access 7 (2019) 39448–39458.

[237] Teleportation in VirtualBox, https://www.virtualbox.org/.

[238] Network functions virtualisation, an introduction, benefits, enablers, challenges & call for action, http://portal.etsi.org/NFV/NFV_White_Paper.pdf.

[239] VXLAN, https://www.cisco.com/c/en/us/products/collateral/switches/nexus-9000-series-switches/white-paper-c11-729383.html.

[240] M. Wang, B. Cheng, J. Chen, Poster: A linear programming approach for SFC placement in mobile edge computing, in: The 25th Annual International Conference on Mobile Computing and Networking, MobiCom 19, Association for Computing Machinery, New York, NY, USA, 2019.

[241] Q. Wang, S. Guo, J. Liu, Y. Yang, Energy-efficient computation offloading and resource allocation for delay-sensitive mobile edge computing, Sustain. Comput.: Inform. Syst. 21 (2019) 154–164.

[242] M. Wang, P.P. Jayaraman, R. Ranjan, K. Mitra, M. Zhang, E. Li, S. Khan, M. Pathan, D. Georgeakopoulos, An overview of cloud based content delivery networks: Research dimensions and state-of-the-art, in: A. Hameurlain, J. Küng, R. Wagner, S. Sakr, L. Wang, A. Zomaya (Eds.), Transactions on Large-Scale Data- and Knowledge-Centered Systems XX: Special Issue on Advanced Techniques for Big Data Management, Springer Berlin Heidelberg, Berlin, Heidelberg, 2015, pp. 131–158.

[243] T. Wang, H. Ke, X. Zheng, K. Wang, A.K. Sangaiah, A. Liu, Big data cleaning based on mobile edge computing in industrial sensor-cloud, IEEE Trans. Ind. Inf. 16 (2) (2020) 1321–1329.

[244] P. Wang, S. Liu, F. Ye, X. Chen, A fog-based architecture and programming model for IoT applications in the smart grid, 2018, arXiv preprint arXiv:1804.01239.

[245] S. Wang, J. Xu, N. Zhang, Y. Liu, A survey on service migration in mobile edge computing, IEEE Access 6 (2018) 23511–23528.

[246] S. Wang, M. Zafer, K.K. Leung, Online placement of multi-component applications in edge computing environments, IEEE Access 5 (2017) 2514–2533.

[247] F. Wang, M. Zhang, X. Wang, X. Ma, J. Liu, Deep learning for edge computing applications: A state-of-the-art survey, IEEE Access 8 (2020) 58322–58336.

[248] S. Wang, X. Zhang, Y. Zhang, L. Wang, J. Yang, W. Wang, A survey on mobile edge networks: Convergence of computing, caching and communications, IEEE Access 5 (2017) 6757–6779.

[249] Z. Wu, H.V. Madhyastha, Understanding the latency benefits of multi-cloud webservice deployments, SIGCOMM Comput. Commun. Rev. 43 (2) (2013) 13–20.

[250] Z. Wu, C. Yu, H.V. Madhyastha, CosTLO: Cost-effective redundancy for lower latency variance on cloud storage services, in: Proceedings of the 12th USENIX Conference on Networked Systems Design and Implementation, NSDI15, USENIX Association, USA, 2015, pp. 543–557.

[251] Xen, https://xenproject.org.

[252] A. Xie, H. Huang, X. Wang, Z. Qian, S. Lu, Online VNF chain deployment on resource-limited edges by exploiting peer edge devices, Comput. Netw. 170 (2020) 107069.

[253] Xin Li, Chen Qian, A survey of network function placement, in: 2016 13th IEEE Annual Consumer Communications Networking Conference, CCNC, 2016, pp. 948–953.

[254] Z. Xu, W. Liang, A. Galis, Y. Ma, Q. Xia, W. Xu, Throughput optimization for admitting NFV-enabled requests in cloud networks, Comput. Netw. 143 (2018) 15–29.

[255] M. Xu, A.N. Toosi, R. Buyya, A self-adaptive approach for managing applications and harnessing renewable energy for sustainable cloud computing, IEEE Trans. Sustain. Comput. (01) 1, http://dx.doi.org/10.1109/TSUSC.2020.3014943.

[256] B. Yang, W.K. Chai, Z. Xu, K.V. Katsaros, G. Pavlou, Cost-efficient NFV-enabled mobile edge-cloud for low latency mobile applications, IEEE Trans. Netw. Serv. Manag. 15 (1) (2018) 475–488.

[257] W. Yang, C. Fung, A survey on security in network functions virtualization, in: 2016 IEEE NetSoft Conference and Workshops, NetSoft, 2016, pp. 15–19.

[258] S. Yi, Z. Hao, Q. Zhang, Q. Zhang, W. Shi, Q. Li, LAVEA: Latency-aware video analytics on edge computing platform, in: Proceedings of the Second ACM/IEEE Symposium on Edge Computing, SEC '17, Association for Computing Machinery, New York, NY, USA, 2017.

[259] S. Yi, C. Li, Q. Li, A survey of fog computing: Concepts, applications and issues, in: Proceedings of the 2015 Workshop on Mobile Big Data, Mobidata 15, Association for Computing Machinery, New York, NY, USA, 2015, pp. 37–42.

[260] B. Yi, X. Wang, K. Li, S. k. Das, M. Huang, A comprehensive survey of network function virtualization, Comput. Netw. 133 (2018) 212–262.

[261] A. Yousefpour, C. Fung, T. Nguyen, K. Kadiyala, F. Jalali, A. Niakanlahiji, J. Kong, J.P. Jue, All one needs to know about fog computing and related edge computing paradigms: A complete survey, J. Syst. Archit. 98 (2019) 289–330.

[262] A. Yousefpour, G. Ishigaki, J.P. Jue, Fog computing: Towards minimizing delay in the Internet of Things, in: 2017 IEEE International Conference on Edge Computing, EDGE, 2017, pp. 17–24.
[263] W. Yu, F. Liang, X. He, W.G. Hatcher, C. Lu, J. Lin, X. Yang, A survey on the edge computing for the Internet of Things, IEEE Access 6 (2018) 6900–6919.
[264] S. Yuan, Y. Fan, Y. Cai, A survey on computation offloading for vehicular edge computing, in: Proceedings of the 2019 7th International Conference on Information Technology: IoT and Smart City, ICIT 2019, Association for Computing Machinery, New York, NY, USA, 2019, pp. 107–112.
[265] D. Zeng, L. Gu, S. Guo, Z. Cheng, S. Yu, Joint optimization of task scheduling and image placement in fog computing supported software-defined embedded system, IEEE Trans. Comput. 65 (12) (2016) 3702–3712.
[266] Y. Zhang, M. Li, K. Bai, M. Yu, W. Zang, Incentive compatible moving target defense against VM-colocation attacks in clouds, in: D. Gritzalis, S. Furnell, M. Theoharidou (Eds.), Information Security and Privacy Research, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, pp. 388–399.
[267] Q. Zhang, F. Liu, C. Zeng, Adaptive interference-aware VNF placement for service-customized 5G network slices, in: IEEE INFOCOM 2019 - IEEE Conference on Computer Communications, 2019, pp. 2449–2457.
[268] F. Zhang, G. Liu, B. Zhao, X. Fu, R. Yahyapour, Reducing the network overhead of user mobility–induced virtual machine migration in mobile edge computing, Softw. - Pract. Exp. 49 (4) (2019) 673–693.
[269] C. Zhang, X. Wang, Y. Zhao, A. Dong, F. Li, M. Huang, Cost efficient and low-latency network service chain deployment across multiple domains for SDN, IEEE Access 7 (2019) 143454–143470.
[270] L. Zhao, J. Liu, Y. Shi, W. Sun, H. Guo, Optimal placement of virtual machines in mobile edge computing, in: GLOBECOM 2017 - 2017 IEEE Global Communications Conference, 2017, pp. 1–6.
[271] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, J. Zhang, Edge intelligence: Paving the last mile of artificial intelligence with edge computing, Proc. IEEE PP (2019) 1–25.
[272] B. Zolfaghari, G. Srivastava, S. Roy, H.R. Nemati, F. Afghah, T. Koshiba, A. Razi, K. Bibak, P. Mitra, B.K. Rai, Content delivery networks: State of the art, trends, and future roadmap, ACM Comput. Surv. 53 (2) (2020).
[273] Z. Zou, Y. Jin, P. Nevalainen, Y. Huan, J. Heikkonen, T. Westerlund, Edge and fog computing enabled AI for IoT-an overview, in: 2019 IEEE International Conference on Artificial Intelligence Circuits and Systems, AICAS, IEEE, 2019, pp. 51–56.

Yaser Mansouri is a researcher with the Centre for Research on Engineering Software Technologies (CREST) at the University of Adelaide working on a project funded by DST Group. Yaser obtained his Ph.D. from Cloud Computing and Distributed Systems (CLOUDS) Laboratory, Department of Computing and Information Systems, the University of Melbourne, Australia. Yaser was awarded first-class scholarship, International Postgraduate Research Scholarship (IPRS) and Australian Postgraduate Award (APA) supporting his Ph.D. studies. His research interests cover the broad area of Distributed Systems, with special emphasis on data replication and management in cloud storage services.

M. Ali Babar is a Professor in the School of Computer Science, University of Adelaide. He is an honorary visiting professor at the Software Institute, Nanjing University, China. Prof Babar has established an interdisciplinary research centre, CREST âĂŤ Centre for Research on Engineering Software Technologies, where he leads the research and research training of more than 30 (12 Ph.D. students) members. He also leads a theme, Platforms and Architectures for Cybersecurity as Service, of the Cyber Security Cooperative Research Centre (CSCRC), which is one of the largest Cyber Security initiative in Australia. Prof Babar has authored/co-authored more than 240 peer-reviewed publications through premier Software Technology journals and conferences. Apart from his work having industrial relevance as evidenced by several R&D projects and setting up a number of collaborations in Australia and Europe with industry and government agencies, his publications have been highly cited within the discipline of Software Engineering as evidenced by his H-Index is 48 with 9643 citations as per Google Scholar on January 16, 2021. This level of citations is among the leading Software Engineering researchers in Aus/NZ. In the area of Software Engineering education, Prof Babar led the UniversityâĂŹs effort to redevelop a Bachelor of Engineering (Software) degree that has been accredited by the Australian Computer Society and the Engineers Australia (ACS/EA). He coordinated both undergraduate and postgraduate programs of Software Engineering at the University of Adelaide for six years. Prior to joining the University of Adelaide, he spent almost 7 years in Europe (Ireland, Denmark, and UK) working as a senior researcher and an academic. Before returning to Australia, he was a Reader in Software Engineering with the Lancaster University.