Information Systems xxx (xxxx) xxx

Contents lists available at ScienceDirect

Information Systems

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

The convergence and interplay of edge, fog, and cloud in the AI-driven Internet of Things (IoT)

Farshad Firouzi [a,*], Bahar Farahani [b,*], Alexander Marinšek [c]

[a] Electrical and Computer Engineering Department, Duke University, United States of America
[b] Cyberspace Research Institute, Shahid Beheshti University, Iran
[c] ESAT-DRAMCO, KU Leuven, Belgium

ARTICLE INFO

Article history:
Received 5 July 2020
Received in revised form 19 June 2021
Accepted 25 June 2021
Available online xxxx
Recommended by Gottfried Vossen

Keywords:
Internet of Things (IoT)
Cloud Computing
Fog Computing
Edge Computing
Mobile Computing
Edge-Fog-Cloud
Cloud IoT
Cloudlet
Offloading
Resource Management
Service Placement
Privacy-Preserving Machine Learning
Security and Privacy
Healthcare IoT
Case Studies

ABSTRACT

The Internet of Things (IoT) tsunami, public embracement, and the ubiquitous adoption of smart devices are affecting virtually every industry, directly or indirectly. The success of the current and future landscape of IoT and connected devices requires service provision characterized by scalability, ubiquity, reliability, and high-performance, among others. In order to achieve this attribution, the integration of IoT and Cloud Computing (CC), known as cloud IoT, has emerged as a new paradigm providing advanced services specific to aggregating, storing, and processing data generated by IoT. While the convergence of IoT and Cloud brings opportunities, it suffers from specific limitations such as bandwidth, latency, and connectivity. The increasing need for supporting interaction between cloud and IoT led to Edge and Fog Computing (FC) in which computing and storage resources are located not only in the cloud but also at the edges near the source of data. The hierarchical and collaborative edge–fog–cloud architecture brings tremendous benefits as it enables us to distribute the intelligence and computation – including Artificial Intelligence (AI), Machine Learning (ML), and big data analytics – to achieve an optimal solution while satisfying the given constraints e.g., delay-energy tradeoff. Due to the hierarchical, cross-layer, and distributed nature of this model, achieving an osmotic and effective convergence of IoT, edge, fog, and cloud computing requires overcoming many challenges with respect to design and implementation, as well as deployment and evaluation. This paper provides a comprehensive insight into the edge-fog-cloud computing paradigm by providing a blend of discussions on all important aspects of the underlying technologies to offer opportunities for more holistic studies and to accelerate knowledge acquisition. To gain a deep understanding of edge–fog–cloud, we will begin this paper by providing an in-depth tutorial and presenting the main requirements, state-of-the-art reference architectures, building blocks, components, protocols, applications, and other similar computing paradigms, including their similarities and differences. Following this, a holistic reference architecture for edge–fog–cloud IoT is presented and the major corresponding design and deployment considerations (e.g., service models, infrastructure design, provisioning, resource allocation, offloading, service migration, performance evaluation, and security concerns) are discussed. Next, we will take a look at the role of privacy-preserving, distributed, and collaborative analytics as well as the interaction between edge, fog, and cloud computing. Finally, we will overview the main challenges in the field of edge–fog–cloud computing that need to be tackled to realize the full potential of IoT.

1. Introduction

The Internet of Things (IoT) is like a living entity that constantly evolves and touches almost every aspects of our lives from in-home appliances to manufacturing robots by bringing people, processes, data and things/objects together in previously unimagined ways [1,2]. On the other hand, Cloud Computing (CC) provides elastic services and virtually unlimited storage and computing capacity on-demand. Although the two worlds of IoT and cloud computing are distinct and have evolved independently, their elements are often complimentary of one another. Thereby, over the past few years, the field has witnessed the convergence of these two technologies, i.e., cloud–IoT paradigm, enabling vast opportunities to drive exciting new applications and services [3–6]. A large portion of the computing needed by IoT devices, such as manufacturing control, virtual reality, or

* Corresponding authors.
E-mail addresses: Farshad.firouzi@duke.edu (F. Firouzi), b_farahani@sbu.ac.ir (B. Farahani).

https://doi.org/10.1016/j.is.2021.101840
0306-4379/

augmented reality, require prompt processing that is latency- and context-sensitive [7]. The integration of Edge Computing (EC) and Fog Computing (FC) into cloud–IoT provided the next big leap disrupting both current and future IoT solutions by reducing latency for critical applications and better handling the massive deluge of data generated by IoT devices. However, the interplay between fog/edge and cloud computing is critical for providing further possibilities in IoT applications, particularly for those relying on Artificial Intelligence (AI) and Machine Learning (ML) [7–14].

An ever-expanding number of devices and growing traffic on the IoT is creating substantial Internet capacity and service challenges. This makes it arduous to handle the time and context-sensitive needs of IoT applications using only cloud computing [7,8,15]. Therefore, computing models are now moving away from centralized models to distributed edge computing paradigms. There are a variety of computing models emerging, such as Mobile Edge Computing, Cloudlet, FC, and Transparent Computing, aimed at using distributed resources at the edge of networks to support time and context-sensitive IoT services. Fog and edge computing are facilitated by network infrastructures that maintain storage and computing not only in the cloud, but also at the edge. Indeed, Edge/Fog computing concept has been introduced to bridge the gap between the cloud and endpoint IoT devices. Note that the characteristics of FC suggest it was not created as a cloud computing replacement because the cloud provides nearly limitless computing power and storage thanks to the economies of scale. Therefore, FC is occasionally referenced as a model that expands cloud to the network's edge. There are several applications (e.g., Smart Grid, Healthcare, Industry 4.0) for which both fog and cloud computing are advantageous. For example, Smart Grid takes the results of big data's preprocessing completed via FC and transmits the data to the cloud for more complex and detailed analysis that is not time sensitive [7,16,17].

Combining the capabilities of edge servers, the cloud, and end devices creates a hierarchical IoT paradigm (i.e., edge–fog–cloud) to enhance IoT system functioning and improve resource usage and Quality of Experience for services. Instead of considering edge, fog, and cloud as separated components, many IoT applications require each component to work in harmony in order to provide dependable services with varying time and location requirements. The hierarchical edge–fog–cloud enables artificial intelligence activities – including machine learning, big data analytics, and decision making – occurs at the edge at or near the source of data or depending on the situation in the fog or even in the cloud infrastructures to able to support complex computations at a distance from smart IoT things. This flexibility of edge–fog–cloud in terms of latency versus computation implies that this hierarchical model is capable of supporting AI-enabled IoT applications in an efficient and scalable way [7,8,15]. However, the distributed nature of edge–fog–cloud architecture presents several challenges (e.g., task offloading, service placement, security, and privacy) that must be addressed. Edge–fog–cloud paradigm is generally considered very heterogeneous and dynamic because a diverse array of devices makes up the total computing power [18]. The devices involved include wearable devices, sensors, smartphones, vehicles, gateways, base stations, servers, and other network nodes with expansive functionality. This can be even intensified with mobile devices impacting resource locations as well as network traffic [7]. Another major challenge is that there is no consensus towards any reference model or best practices that specify how edge–fog–cloud should be utilized. Thereby, in this paper, we first summarize and categorize the attempts have been made in this domain. Next, we provide a very detailed insight into edge–fog–cloud interplay and the related computing paradigms. Then, a holistic reference model is proposed. Finally, we discuss the corresponding challenges (e.g., interplay among components, offloading, service placement, distributed and privacy-preserving collaborative ML) and relevant research topics to better support IoT applications.

The remainder of this paper is organized as follows. Section 2 discusses the related work. Section 3 overviews the state-of-the-art edge/fog computing architectures and presents the corresponding fundamentals and the recent advancements. Next, in this section, a holistic reference architecture for edge–fog–cloud IoT model is proposed. In Section 4, we discuss and compare the related paradigms and technologies. Section 5 addresses the design considerations as well as the most important topics of the presented edge–fog–cloud IoT from service provisioning to offloading techniques. Section 6 presents what challenges and barriers have to be tackled to grow further. Finally, Section 7 concludes the paper.

2. Related work

There are some existing related studies in the area of fog and cloud computing presenting definitions, architectures, application scenarios and certain challenges (see Table 1) [16,17,19–28]. Notably, Martinez et al. [19] systematically discuss the implementation aspects required to design and build a holistic resource management infrastructure for large scale fog computing to support IoT applications. Azam et al. [27] present a taxonomy of recent offloading schemes as well as the required middleware technologies enabling offloading in cloud–IoT. Yousefpour et al. [16] present a complete survey discussing several important aspects of fog/edge computing from recent architectures to related paradigms, resource management, provisioning, task allocation, open issues, and challenges. Naha et al. [20] present the state-of-the-art and identify and discuss fog computing requirements, including existing fog computing architectures and the related components, the taxonomy of fog computing, resource allocation and scheduling, fault tolerance, and simulation tools. Security and privacy aspects of edge/fog computing is discussed in [28]. A holistic and in-depth analysis of requirements and different solutions of fog computing for IoT applications is conducted in [21]. Finally, an overview of service placement problem in fog and edge computing is presented in [29]. Unfortunately, most of the existing works only focus on a specific topic and characteristic (e.g., applications, standardization activities, security, and privacy challenges, offloading, and requirements of infrastructure), so there is a lack of study to cover all important aspects of edge–fog–cloud computing for IoT use cases. To the best of our knowledge, none of them comprehensively addressed various aspects of privacy-preserving, distributed machine learning, and collaborative analytics to harvest data deluge created by IoT devices. Indeed, the interplay of edge–fog–cloud, how AI methods can be deployed in an efficient and scalable way considering the privacy issues of IoT data owners, and the interaction between fog analytics and cloud analytics have not been discussed deeply. The main contributions of this paper include:

- This paper provides a detailed tutorial and presents a comprehensive and critical evaluation of existing solutions, models, and architectures from edge to fog and cloud by considering the requirements of IoT applications. In addition, a holistic reference model and several guidelines are provided for building effective edge–fog–cloud solutions addressing IoT applications.
- This paper acquaints the readers with the state-of-the-art topics covering standards, protocols, design principles, reference architectures, the underlying technologies, pillars, components, the interplay between cloud and edge/fog, related computing paradigms including technical similarities and differences, distributed and collaborative analytics, as well as open issues to unleash new research opportunities.

- This holistic paper is designed to fill the gap and to help the industry and research communities synthesize their effort by providing an invaluable reference addressing all important aspects and challenges in architecting and engineering edge–fog–cloud IoT solutions from resource management to resource provisioning, resource allocation, service placement, task offloading, security and privacy, performance evaluation and the related simulation/emulation tools, data analytics, and applications and use cases.
- This paper presents the most challenging open issues, research opportunities, and directions for research in edge–fog–cloud IoT.

3. Edge–fog–cloud architecture

In this section, we analyze the IoT requirements from the perspective of edge–fog–cloud architecture and discuss the existing architectures as well as the most common applications. In addition, we present a reference architecture for edge–fog–cloud and study the corresponding components as well as design aspects and considerations.

3.1. IoT requirements

IoT require cloud and fog computing to meet the demand of the emerging applications. The major IoT requirements are outlined below [8,9,20,21,27]:

- **Collaborative and Distributed Computing**: AI is the key to unlock the potentials of IoT by wringing insights from data more quickly. Due to characteristics of IoT data (i.e., volume, variety, velocity, and veracity), traditional Cloud-based IoT model cannot meet the requirement of emerging IoT applications. Thereby, it is imperative to distribute the intelligence across the entire IoT layers to achieve both high speedup (i.e., less response time) and low data transmission [8].
- **Scalability**: The scalability is the primary need of IoT spread across a wide geographic area with varying density levels. Thereby, IoT relies on different topologies and distributed configurations in order to adapt and scale according to each system's needs [17,20,21,34].
- **Interoperability**: IoT is operating a very heterogeneous environment that requires a variety of devices/systems to communicate and interact with each other. Heterogeneity in complicated systems can impact semantic and technical interoperations. Standardizing how information is described or exchanged in addition to an abstraction layer able to obscure elemental differences is needed to support interoperability [17,21,34].
- **Real-time**: Responding with low-latency to handle real-time interactions is a primary need for IoT applications used in the real world. In a very dynamic scenario, data can change rapidly and data exchanged between the cloud and an IoT system may be inaccurate due to high-latency in interactions. FC is very important in meeting low-latency needs because the cloud and edge cannot interact directly [15,20,21].
- **Data Quality**: Data quality is required by real world IoT applications because it directly impact the system reliability and performance [15,21].
- **Security and Privacy**: IoT requires solutions that can support security, privacy, and safety by closely interconnecting those aspects to one another [1,21].
- **Reliability and Resiliency**: IoT requires that many devices work in a distributed way to accomplish tasks and activities, making reliability an essential requirement. IoT depends on pillars of serviceability, reliability, and availability to address multiple issues. Reliability is required by different layers and components. Hardware, such as sensors, must operate within expected parameters, network elements must communicate reliability to transmit data and exchange messages, and computing nodes must generate anticipated outputs such as data processing [21,34].
- **Mobility**: IoT is adapting to handle highly mobile devices. System knowledge must move freely in data-dense mobile applications. Locating the correct data improves performance, local caching, and data models. In addition, IoT solutions should enable mobile devices to move from one authority region to another without disrupting the system or creating issues [21,34].
- **Location Awareness**: Location awareness can strengthen IoT applications by providing systems with greater consciousness and resilience. Location awareness can also improve sensed-environment knowledge, which enables the system to adapt and improve execution and application quality [21,34].

3.2. Existing architectures

The current FC architectures are layer based. Different researchers suggest fog-assisted IoT architectures holding anywhere from 3 to 6 layers (i.e., one to four fog layers). The most well-known fog architectures include OpenFog, IFCIoT architecture, hierarchical fog model, IoE fog architecture, nervous system-based architecture, and fog network model. Most of the fog architectures found in current literature utilize only one layer fog (i.e., the three-layer IoT architecture) that extends from the IoT device layer to the cloud layer [4,5,19,35–37]. Note that typically fog nodes can be connected to create a grid that handles fault tolerance, load balancing, resilience, cloud communication minimization, and data sharing. This means that nodes must be able to communicate laterally with peers and vertically within the hierarchy [19,38]. Some edge–fog–cloud reference models are organized as hierarchies with several tiers [19,39,40]. In these models data is shared vertically across fog layers but not inside the same horizontal layer. The nodes nearer to the edge are usually concerned with gathering sensor data, controlling actuators or sensors, and normalizing data. Nodes in the layer above are generally concerned with filtering, compressing, and transforming data. These nodes also sometimes handle edge analytics needed for real-time or near real-time processing. Moving further from the network's edge usually involves more complex machine and system learning ability [19]. Some work visualize fog architecture in the shape of a tree with nodes rooted in the cloud [41]. As proposed by [42], rather than depicting hierarchical tiers of fog nodes, fog devices can be organized into clusters with inter- and intra-fog communication ability. Fig. 1 visualizes a few common FC architectures. Deployment of FC can vary from large, interconnected cluster systems to being embedded. The kind of deployment depends on the situation, but basic elements of the architecture are included no matter the type of deployment. The number of layers included also depends on the specific scenario's requirements, including:

- Kind and amount of work required of each layer
- How many sensors are included
- Node capabilities within each layer
- Level of latency among nodes as well as between sensors and actuators

Table 1
Summary of most recent work on Edge–Fog–Cloud.

Reference	Year	Fund. & Def.	Arch. & Related Paradigms	IoT Require-ments	Edge–Fog–Cloud Interplay	Resource Manage-ment	Provi-sioning & Alloca-tion	Offload-ing	Service Place-ment	Security & Privacy	Perfor-mance Evalua-tion	Open Issues & Chal-lenges	AI/ML Tech.	Applica-tions & Use Cases
[19]	2020	x				x	x							x
[27]	2018	x			x			x						
[16]	2019	x	x			x	x	x	x			x		x
[28]	2018	x	x							x		x		
[20]	2018	x	x			x	x		x		x	x		x
[21]	2019	x	x	x										
[29]	2019	x							x					
[30,31]	2017	x		x										x
[32]	2016	x		x	x	x								x
[33]	2015	x												x
This Work	2021	x	x	x	x	x	x	x	x	x	x	x	x	x

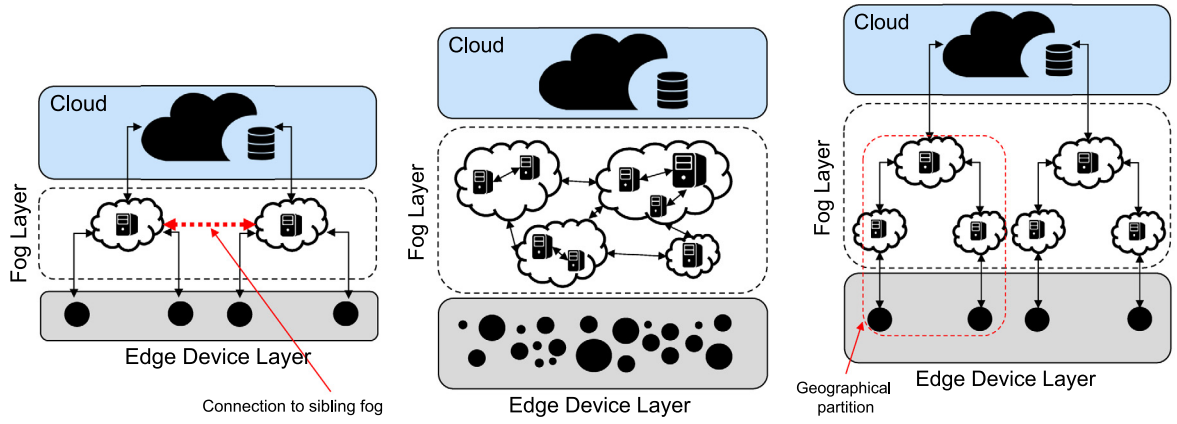

Fig. 1. Examples of Edge–Fog–Cloud architectures: (a) 3-Layer Model; (b) Clustered Model; (c) Tree-based Model.

3.3. Reference architecture

Combining the available architectures and taking into account the frequent FC use cases, we present a hierarchical reference model (see Fig. 2) to assemble a universal computing infrastructure, which can cater to the needs of IoT applications various domains. The reference hierarchical computing model's architecture follows a hierarchical multi-layer pattern of placing edge, fog, and cloud devices each in their corresponding layer. Furthermore, the fog is broken down into additional layers to reflect the distribution of FNs throughout the network infrastructure. Note that FN can communicate with each other vertically and horizontally depending on the load of the system and the requirements of the application. This is usually controlled and managed by Fog Orchestration Nodes (FON). In other words, FONs create the control layer of the fogs (or fog clusters). Note that depending on the application, fogs can also include Fog Gateway Nodes (FGN), gateways or access points serving as entry points to fogs. In addition, fogs also include Fog Computing Nodes (FCN) – general computing nodes – to execute tasks assigned by FONs. Increasing the flexibility of the edge–fog–cloud architecture, the model can designed to further enable edge-device-driven resource orchestration. By enabling an edge device to invoke Service Placement (SP) on its own, it can avoid starvation in multi-user scenarios, where only a single device requests access to a specific service. Edge devices are also allowed to place custom executable tasks at FNs in the form of user-space containers. That way, an edge device may, for example, host part of a Convolutional Neural Network (CNN) itself, while the remaining layers are bundled together in a container and execute by the nearby fog. Giving edge devices the ability to migrate services also augments the edge–fog–cloud model's mobility support by allowing the edge devices to predict potential FN handoffs based on their own movement.

The conceptual reference architecture is made up of a horizontal dimension as well as a vertical dimension [3,7,16,21,43].

- **Vertical Dimension**: This dimension is made of 3 main layers including the cloud, fog, and edge. Each layer contains different types of nodes required to handle component tasks. Also note that fog layer itself can also consist of several sub-layers.
- **Horizontal Dimension**: According to [21,43], this dimension captures the following perspectives:
  - **Node Perspective**: This perspective includes accelerators, computation, sensors, actuators, and storage equipment. Fog servers, gateways, or devices can stand alone or be connected to the IoT [21,43].
  - **Communication Perspective**: The network provides flexibility, scalability, and availability required of communication processes as well as the QoS needed to handle delivery of low latency or vital data. Based on the specific situation, fog nodes usually exist in a network component like a router, access point, or gateway [21,43]. The connectivity model depends on the node's location and function. For example, fog nodes used in an industrial setting may need to support Controller Area Networks(CAN) buses to communicate with lower tier processes or applications [21, 43]. When using IoT applications, wireless connections

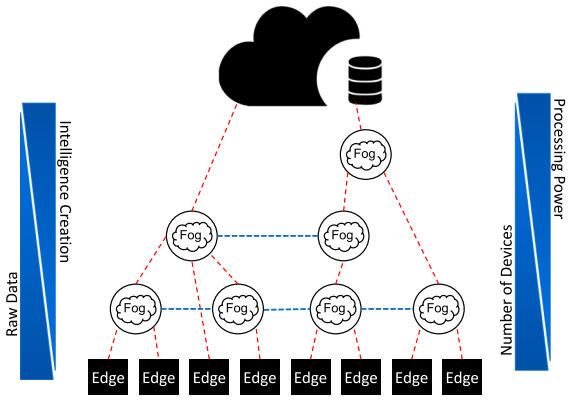

Fig. 2. An illustration of the proposed edge–fog–cloud computing hierarchical architecture.

are most important to downward communication between sensors and nodes. It can also be utilized for interconnections between the cloud and fog nodes or node-to-node connections. Wireless support of nodes depends on the node's location in the architecture hierarchy, data throughput, mobility, data velocity, coverage, the environment, spectrum licensing, and energy sources [21,43].

- **Control and Management Perspective**: This perspective captures lifecycle management, registration, provisioning, automated discovery, offloading, load balancing, task placement, task migration, and resource allocation, among others [21,43].
- **Data Processing and Analytics (AI/ML) Perspective**: This perspective discusses how to effectively process, analyze, and manipulate the tsunami of multi-scale, multi-modal, distributed and heterogeneous IoT data [21,36].

3.4. Edge–fog–cloud interplay

The widespread adoption of cloud computing has led to the reduction on edge device local storage demands and it has enabled the edge devices to offload computationally intensive tasks. Fog computing on the other hand has enabled the edge devices to offload tasks with lower transmission latency and greater resilience in case of heavy traffic on the network backhaul. Neither is perfect, yet their carefully coordinated interplay enables edge devices to reap the benefits of both worlds.

- **Reduced Network Load**: Handling data streams at the edge of the network by bringing computing power closer to data sources evenly distributes the load across the network, and ultimately reduces the burden on the system's backhaul. Consequently, the need for increasing backhaul data rates is avoided and the data links to the far end of the network are reserved for the services that need them.
- **Latency-aware Computing**: Network load reduction is not the only benefit of redistributing processing resources across the network, as edge devices also enjoy faster task processing times, the consequence of reduced data propagation times. Unsurprisingly, FC greatly contributes to the increase in the Quality of Service (QoS), and makes it possible for latency-critical applications such as immersive multimedia to start conducting task offloading without violating their stringent latency constraints.
- **Native Support for Mobility**: Bringing resources closer to edge devices also means the network can respond to the mobility of the users more promptly and precisely. When Fog Nodes (FNs) are able to communicate directly with one another (horizontal communication), task offloading and request handling are able to continue even in high mobility scenarios, such as vehicles traveling on the highway.
- **Providing Context**: Since resources are positioned in close vicinity to the end users, they can provide them with specific content related to that geographical area. Furthermore, if the FNs are aware of their absolute position, they can provide additional inherent services to the users, such as precise localization.
- **No single Point of Failure**: The distribution of resources throughout the network also means that if a certain data link is disabled because of maintenance or if an FN goes offline because of a cyber attack, the remaining FNs can take over the workload while the network continues to function uninterruptedly.
- **Longer Battery Life**: Because FC data processing delays are low, edge devices can more heavily rely on task offloading. By doing so they reduce their own power consumption and increase their battery life, allowing them higher long-term autonomy.
- **Lower Energy Consumption**: Given that most of the data gets processed close to its source, multi-hop long distance data transmission is avoided, bringing down the total energy consumption of the network. Moreover, since the power loads are more evenly distributed, their power demands are easier to meet using the existing power grid infrastructure and renewable energy sources.
- **Support for Heavy Loads**: With the help of CC, the edge–fog–cloud computing model can always rely on the cloud possessing enough resources to dwarf even the most powerful FNs. With this in mind, an overloaded FN can forward a daunting computational task to the resource-rich cloud, enabling its execution at the expense of longer network propagation times.
- **Infinite Storage**: Avoiding the need to integrate additional storage capacities into the space-constrained edge devices and FNs, the edge devices can again rely on the cloud's ability to almost infinitely expand its resources and meet their storage demands.

3.5. Edge–fog–cloud computing applications

Edge–Fog–Cloud computing has numerous applications. In the following section, we briefly overview a few of them:

- **Healthcare**: Edge–Fog–Cloud computing can be utilized to overcome big data obstacles within the healthcare industry. These challenges include the geographical location of providers and lack of interoperability, strict security and privacy rules governing patient health data, and transforming big data into smart data. Edge–Fog–Cloud computing also supports the use of real-time sensors in patient care [1].
- **Smart Transportation Systems**: Smart transportation applications can also utilize Edge–Fog–Cloud to monitor public buses, trains, or subways in order to provide real-time information about where the transportation is located or any potential delays. Traffic flow can be optimized using smart traffic lights, cameras, and vehicle sensors. Smart cars can calculate ideal routes and avoid potential collisions by working with Edge–Fog–Cloud and road-side infrastructure to process current traffic conditions. Vehicular Ad-hoc Network (VANETs) is another main application that enable vehicles, infrastructure, and roadside units to work together [44,45].

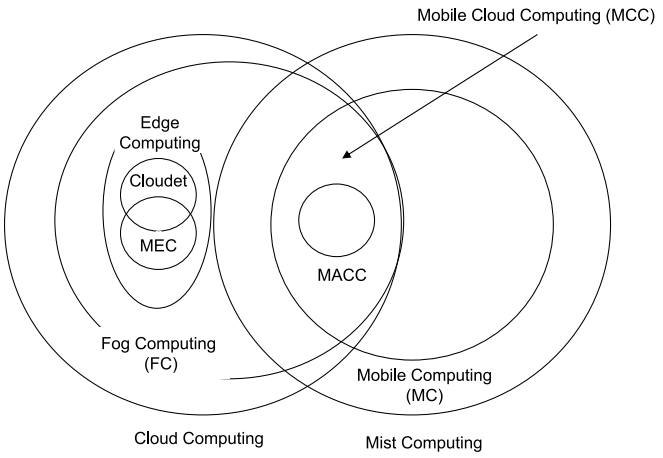

Fig. 3. Comparison of Fog Computing (FC) and its related computing paradigms.

- **Smart City**: Smart city applications must ingest and analyze sensor data in real-time. Edge–Fog–Cloud could assist with smart streets, mapping of noise pollution, support drainage networks, and many other interesting applications [20].
- **Smart Grid**: Smart grids utilize two-way electricity and information flow to provide automated energy delivery. Smart grids also provide transparent distribution where customers and providers can monitor production, use, and prices in near real time. While smart grids can be developed with cloud computing, there are low-computation and communication needs involved in creating a low-latency, highly-private grid. These issues can be addressed with Edge–Fog–Cloud while providing fault tolerant services, data privacy, and scalable real-time services [45].
- **Multimedia**: Because of its ability to handle real-time processing, Edge–Fog–Cloud is capable of delivering processing to diverse media including video streaming, augmented reality, and gaming. In addition, video surveillance system may employ Edge–Fog–Cloud for facial recognition which could reduce the response time of authorities in the case of an emergency [38].

4. Related paradigms and technologies

4.1. Existing architectures

The frequently debated and most promising architectures are summarized in the following subsections (See Fig. 3).

4.1.1. Software-Defined Networking (SDN)

SDN is a category of technologies for network management decoupling the data plane (i.e., forwarding process of network packets) from the network control plane (i.e., routing process). At the heart of any distributed computing architecture lay a set of mechanisms for rerouting, forwarding, and capturing user requests. Facilitating these functionalities, SDN is a cornerstone of distributed computing, making it possible to smoothly and seamlessly extending computational resources to the edge. In conventional internet networks the data and control planes are tightly coupled, and the presence of statically placed middleboxes, such as firewalls, makes the network extremely rigid. Instead, SDN splits the two planes to allow for a separation between forwarding and control logic. In a software-defined network, routers, switches, wireless access points, and other forwarding devices, are represented as basic forwarding hardware, while the control logic is offloaded to a nearby orchestrator [46]. Depending on the desired results and the available hardware, the control logic can be centralized or distributed, fine- or coarse-grained, and reactive or proactive. The control layer communicates with the data plane through a set of standardized APIs, while in some cases an additional plane is added above the control devices. The management plane's role is to further monitor the network's behavior and impose enriched control mechanisms [47]. With SDN, edge devices no longer need to worry about choosing the optimal destination for task offloading. Likewise, cumbersome network operations, such as IP table updating, are avoided. Instead, it is the SDN orchestration framework's responsibility to meet requirements of the edge device requests based on the current state of the network. Therefore, the shortest path from data source to data consumption is dynamically established. The routing efficiency and resilience are further enhanced when the orchestration framework has real-time insight into the available network resources and their locations. This is the case if the orchestrator resides at the core of the network, or when multiple SDN orchestrators in lower layers are able to communicate with each other and exchange information about the state of the network in their vicinity. Granting SDN orchestrators both control and an overview of the entire network, they can repeatedly establish the best routes between edge devices and services, regardless of user mobility and the redistribution of computing resources within the network [48].

4.1.2. Multi-access Edge Computing (MEC)

One of 5G's main advantages over previous mobile networks is its ability to process data generated by edge devices at or near to the edge of the mobile network, significantly contributing to latency assurance and lowering the load on the network's backhaul. With the use of a standardized application programming interface (API) and by employing network slicing, MEC provides a consistent means of resource access to service developers. On the other hand, its support for data caching, single-node processing, and distributed computing serve the needs of edge devices ranging from vehicles exhibiting high mobility to manufacturing floor machines imposing strict latency constraints [49]. In contrast to FNs, MEC servers are not comprised of consumer devices, but are instead represented by a set of computing nodes purposefully installed at the network edge by a mobile network operator [50]. Although this means the commissioning of MEC servers may not be as fast paced as that of FNs, MEC's strict standardization and its integration into the remaining network architecture makes it possible to considerably cut down on the time needed for application development. For instance, MEC servers will come pre-equipped with built-in services, accessible through a unified set of APIs and granting applications access to radio network information, bandwidth management and location data [51]. The MEC servers reside one layer below the packet core of the mobile network, directly at the 5G radio access network (RAN) or at an aggregation point such as a baseband unit (BBU), further towards the core network. Additionally, a MEC server can complement an LTE eNodeB base station (BS), a 3G radio network controller (RNC), or it can provide for the needs of Wi-Fi users through a Wi-Fi access gateway (WAG) or its equivalent [52]. Since MEC relies on mobile networks for communication purposes, it offers high levels of coverage; yet, having its roots in mobile communications infrastructure also means the deployment of MEC servers is highly dependent on the investments in mobile network infrastructure. At the same time, the pursuit for 5G standardization and recent full-blown 5G infrastructure roll-out announcements make MEC the most mature distributed computing architecture for the time being.

4.1.3. Mobile Cloud Computing (MCC)

Offloading computationally demanding tasks directly to the cloud, MCC is one of the first widely adopted architectures for computation offloading. It gained major traction in the early 2010s when most edge devices, such as smartphones and eHealth wearables, were still not capable of executing AI applications in real-time [53,54]. The procedure of offloading and processing a task is similar to that of FC, with one major drawback: latency. Whenever an edge device decides to offload a task, the data gets relayed all the way to the cloud, residing at the core of the network. There the task is processed before being sent back to the edge device. Thus, although potent data centers process the offloaded task in negligible time, the end-to-end communications latency – in the range of 10 s or even 100 s of milliseconds – exceeds the budget of most applications.

4.1.4. Dew Computing (DC)

Closely tied to cloud computing, DC is a paradigm mostly associated with secure and reliable data storage. The aim of DC is to extend the functionalities of the cloud onto edge devices, where services can run both independently and in collaboration with the cloud [55]. Therefore, as long as a connection between the edge device and cloud exists, the two cooperate to provide rich functionality. When the link is broken, the execution of the service on the edge device continues to run uninterruptedly, although with less functionality. Illustrative examples are Dropbox and Git: a user always has access to their local data storage, while a connection to the cloud is needed for data synchronization. These principles also form the foundation of FC. On its own, DC suffers from the same shortcomings as MACC and is not suitable for latency-sensitive applications.

4.1.5. Cloudlets

Regarded as *data centers in a box*, cloudlets offer an alternative to MCC by moving a miniature version of the cloud-based data center closer to edge devices [56]. They make up a potent mixture of decentralized computational devices widely dispersed across the network infrastructure, also referred to as *edge clouds* [57]. Similarly to cloud computing concepts, cloudlets leverage the abstraction offered by Virtual Machines (VMs) to accommodate edge services and tasks offloaded by edge devices [57]. Since cloudlets are closely bonded to the concepts commonly found in CC, while possessing adequate resources for continuous VM orchestration, they have not been subject to any rigorous standardization or other formal measures of globally unifying their architecture. Nonetheless, their long-standing presence makes cloudlets a relatively mature architecture. Being closely associated to data centers and CC, a cloudlet is usually comprised of purpose-made equipment, dispatched and set up by a large service provider. Because they possess adequate computational capabilities to serve a multitude of edge devices, a single cloudlet can afford to serve multiple households, office buildings, or research institutions at once. Furthermore, their size makes them impractical for placement in close proximity to the edge devices. Unfortunately the slightly more distant placement, combined with vague standardization, mean cloudlets are somewhat limited in providing context to edge devices. Nevertheless, a bottom-up approach is not uncommon either, and a cloudlet may take the form of a resource-rich computer or even a cluster of single board computers such as the Raspberry Pi [58] intentionally placed in close proximity to a group of edge devices. This brings down the cost of a single cloudlet, but the burdening task of having to set up each individual cloudlet as a distributed computing platform remains.

4.1.6. Mobile Computing (MC)

In MC, each edge device with sufficient computing capabilities attends to the processing of its own tasks. A common example are smartphones. With the evolution of their processing performance [59], these advanced edge devices are able to instantaneously process and act on the integrated sensors readings. The multi-gigahertz, multi-core processors found in today's smartphones enable them to run complex ML algorithms [60] and avoid having to relay computationally intensive tasks to external computing nodes. Similar trends of increasing processing capabilities also apply to other consumer-grade edge devices, such as augmented reality (AR) headsets and smartwatches. They often boast roughly the same internal workings as smartphones. Nonetheless, the mentioned edge devices cannot compare with the capabilities of even the smallest of data centers. At least not on their own.

4.1.7. Mobile ad hoc Cloud Computing (MACC)

In the so called *ad hoc mobile clouds* or *mobile ad hoc networks*, the infrastructure is built up of individual edge devices. They operate independent of a central orchestration platform while working on achieving a common goal. By forming ad hoc clusters, the groups of edge devices are able to mimic a larger processing platform such as a MEC server or an FN and process tasks offloaded to them by other edge devices [61]. The mobility of edge devices is reflected in the ad hoc clusters, since mobile edge devices are constantly joining and leaving individual clusters. Moreover, when the edge devices jointly move in the same direction, so does the entire computational node (ad hoc cluster) [62]. The MACC architecture is unique as the devices within the cluster may communicate via any means of wireless communication (Wi-Fi, Bluetooth, ZigBee, etc.) [16]. MACC is indispensable in cases when a large group of edge devices gets cut off from the network. Examples include large venues such as sports competitions and concerts. In such cases the network infrastructure most often fails under the high load cause by the numerous edge devices. To continue serving the demands of computationally burdening applications such as those of AR, the cut-off devices can form an infrastructure of their own. Note that in cases when a distributed processing architecture is readily available, however, forming *ad hoc mobile clouds* would add complexity to the network and typically offer little reward.

4.1.8. Extreme edge (mist computing)

While in MACC edge devices join one another to process a computationally intense task, resource rich extreme edge devices [63] may decide to individually contribute their processing power to nearby edge computing nodes [64]. For instance, stationary extreme edge devices such as parked vehicles allow the MEC server to offload part of the workload onto them, while they trade their processing time for profit [65]. Another example are smart thermostats. They collect data from the likes of temperature and movement sensors, interpret it, and respond to events through actuators, such as air conditioning devices. Such cases, where data processing is carried out by stationary extreme edge devices, are also referred to as *Mist computing* [66]. Extreme edge and mist computing device can exist alongside the FC architecture. They are represented by a thin layer of constrained computing devices, wedged between the sensing devices and the first layer of the fog architecture. Their main trait is high context awareness.

4.1.9. Transparent Computing (TC)

Emphasizing flexible offloading from the cloud to the edge devices, TC lets user equipment free itself of the underlying OS, libraries, and applications. All of these software components are instead migrated to the cloud. This makes it possible for the

stripped down edge devices to request and migrated a particular software configuration on demand. The result is an edge device fully optimized for running a particular service. Given it can now dedicate all of its resources for processing tasks associated with a single service, the edge device no longer has to rely on task offloading [67]. Should the edge device wish to switch its focus to another service, it does so by initiating the same sort of migration procedure as previously mentioned. For more meticulous applications, the edge device can also decide to migrate only specific service parts, frequently altering its configuration [68]. The procedures associated with TC can be viewed as the inverse to those of FC. Instead of the edge device offloading tasks to the network, the network offloads software configurations onto the edge device. This makes TC particularly useful in environments where a reliable data link to the network cannot be established. For example, a UAV for remote sensing that is used both by the search and rescue teams and firefighters. In order to conduct either forest fire or missing person detection in real-time, the necessary configuration associated with each use case is swiftly loaded onto the device before takeoff. Note that in urban areas, where signal coverage is abundant, task offloading is still preferred since it considerably reduces the complexity of the edge devices.

4.1.10. Gateway-Mediated Edge Connectivity (GEC)

Elsewhere in the realm of IoT, certain edge device usage patterns are starting to make space for additional simplifications within the multi-layer architecture. A prominent example can be found in the industrial IoT (IIoT), where the three-tier fog architecture is further sliced into multiple horizontal layers according to the locations of the FNs and the corresponding propagation delays. The fog layer closest to the edge is envisioned to support the inclusion of edge gateways or hubs, which act as endpoints of the WAN and, at the same time, isolate the local area network (LAN) from the WAN. The edge gateway therefore takes the form of the single entry point into the WAN, while it represents a local management and data aggregation platform for the edge devices. With the added gateway-mediated edge connectivity (GEC) abstraction [69], the complexity of the architecture is broken down, as the edge devices may scale up in numbers without directly affecting any other FN, except the nearby edge gateway. The GEC pattern is particularly suitable for IIoT applications since edge devices tend to stay within the reach of the same edge gateway, which can thus rely on clustering the connected devices together, without the need for frequent regrouping. Note that The GEC paradigm is less suitable for serving highly mobile edge devices on lively streets or busy supermarkets. These would trigger high amounts of regrouping overhead, consuming precious time and resources that could be allocated to task processing instead.

4.2. Summary and comparison

To give a better overview of the architectures from Section 4.1, five performance metric were selected and associated with each architecture.

(1) *Infrastructure resilience:* The ability of the architecture to adapt to abrupt changes in the network. FC received a high score as in the event of a FN malfunction, the orchestrator reroutes traffic to the FN's neighbors and precessing can resume. Contrarily, the centralized nature of MCC makes it susceptible to single points of failure.
(2) *Resource availability:* The average amount of resources available per computational node. As small data centers, cloudlets perform well in this aspect. However, since the fog stretches a host of devices, from the network's edge to its core, the average computing capabilities of FNs are also relatively high.
(3) *Proximity to edge devices:* The distance between computational nodes and edge devices, and with it the corresponding time delay. Extreme edge (mist) devices fair best in this aspect since they reside next to the sensing devices. Owing to its presence in APs and similar devices at the network's edge, the fog can achieve latency results that are on a pair with those of mist devices.
(4) *Context awareness:* The ability to provide information about the edge device's environment. Surrounding the edge device with multiple peers, the devices in an *ad hoc mobile clouds* can fuse the information from individual devices and provide accurate descriptions of the environment. Examples include positioning and air quality monitoring. Since MEC also provides localization services, it outscore the fog by a small margin. Although close to the edge devices, the fog has to deal with multi-vendor equipment is less precise in providing context awareness.
(5) *Architecture maturity:* The current state of the architecture. Cloudlets are based on data centers and they are already deployable. MEC is closely bounded to 5G, meaning the latter also is paving the way for widespread MEC adoption. Since FC is a relatively new paradigm and it is not connected with any emerging technological standard, it is for the time being still not widely available.

The evaluation of the architectures based on the performance metrics is summarized in Fig. 4. For the purpose of clarity, only FC and its main competitors were includes. The remaining architectures were omitted in correspondence to the following assumptions:

- DC was left out since it represents only a part of the functionality that other architectures support.
- GEC represents a specific subset of FC. It would score similarly to the fog, albeit, slightly worse.
- MCC was not included, as it would score similar to cloudlets, only slightly more in terms of resources and maturity, while it would fair worse in terms of resilience, proximity and context.
- Although extreme edge (mist) devices would obtain the highest score for context awareness, their resource availability is negligible compared to other architectures. MC and TC were left excluded for similar reasons.

5. Topics and design considerations in edge–fog–cloud computing

The Edge–Fog–Cloud paradigm covers a broad array of topics. The content contained within the current section aims to shed light on these topics and provide an overview of recent research directions.

5.1. Service models

Service modeling has been the subject of debate before the widespread adoption of FC [70]; however, as the underlying system architecture changes, the service models must adapt to the new infrastructure while maximizing QoS demands. Properly structuring a service determines how well it will fit alongside other services at an FN, how challenging it is to migrate the service from one FN to the other in high-mobility scenarios, and how close the service can be brought to individual edge devices. The following segmentation departs from the findings presented in [29] and adds additional important aspects of service modeling. As an aggregation point of the service structures, Table 2 lists the discussed service models and the research works where they appeared.

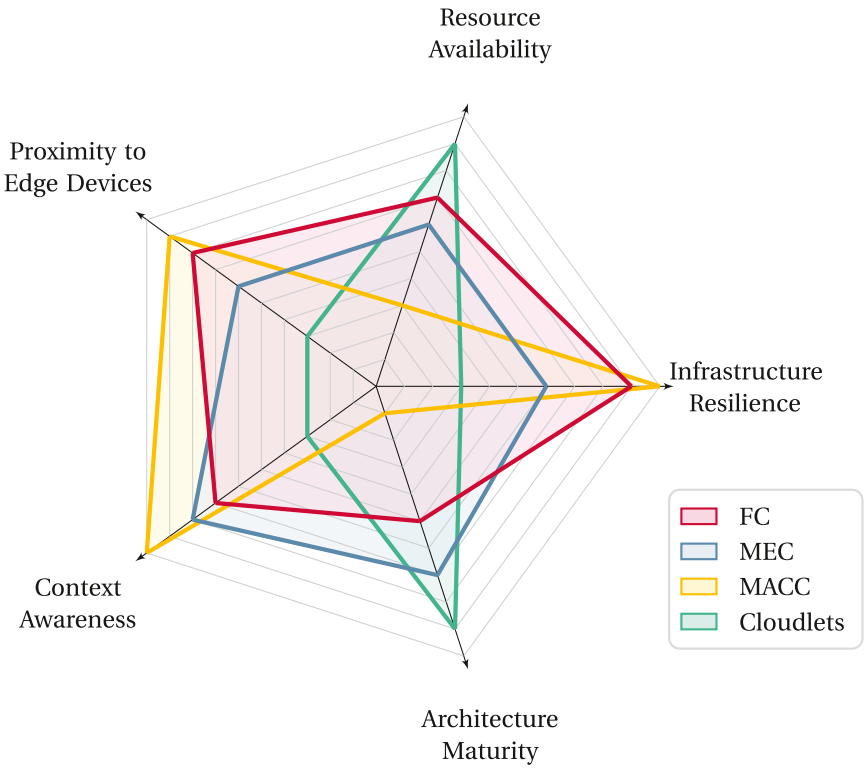

Fig. 4. Comparison of FC, MEC, MACC and cloudlet distributed computing architectures.

Table 2
Service structure.

Service structure	References
Monolithic	[71–78]
DAG	[73,79–81]
Other (distributed)	[71,82,83]

5.1.1. Monolithic services

Modeling services as monolithic components means that each of them is placed at an FN as a single entity. Since the whole service is bundled together, the need for careful coordination and synchronization between dispersed service modules is eliminated. Furthermore, multiple analogous services may be bundled together and nested within a VM which is then replicated and copied to multiple FNs, depending on the requirements of the edge devices [84]. Unfortunately, trying to evenly distribute whole VMs or large service blocks across the edge layer is far more challenging and less exact than when optimizing an service placement (SP) problem where each service is made up of several smaller distinguishable entities.

5.1.2. Set of inter-dependent services

In light of latency reduction, services may be split into several parallelizable modules, where each module attends to a specific computational task, before combining the results to form the final response in timely manner [71]. The parallelizable nature of such services comes into play if an FN also houses a graphics processing unit (GPU). Since all of the parallel processing is done at the same location, reassembling the results is relatively straightforward. However, when the parallelization takes place across multiple FNs, discrepancies in the timeliness of their responses come into play. The time differences are equalized using rigorous time synchronization, which unfortunately increases system complexity and communication overhead.

5.1.3. Services as a connected graph

Services can be seldom dissected into equally-sized entities that enable parallel execution, and the preferred approach is to instead model each service as a set of inter-dependent components, making the service take the form of a connected graph [70] - most commonly a directed acyclic graph (DAG). Modeling large IoT applications as DAGs makes it possible to place individual microservices as close to the individual sensors and actuators as possible.

5.1.4. Services in light of elasticity and multi-user support

When the edge layer infrastructure is overloaded, violations of the service requirements, such as maximal allowed latency, are bound to occur. If an edge device requires a real-time result it is therefore better to reduce the quality of the processing, while still complying with the upper latency limits. Typically, a service provides several quality tiers which are dynamically selected for each user request, depending on the observed latency. An example are image processing and video rendering where the individual tiers represent different image resolution pre-scalers [85,86]. Where the service is dependent on a multitude of sensor data, the elasticity is reflected in the amount of sensor data used for processing a single request [87]. A relevant feature that the SP optimization algorithms must consider is, whether a service allows resource sharing amongst different users. Single user services are suitable for stateful applications, such as those common for virtual reality (VR) [76], or applications demanding high data privacy. However, the overheads caused by service virtualization add up, yielding sub-optimal resource utilization. Keeping track of their own application state, multiple edge devices can simultaneously access a single instance of a multi-user service — decreasing the load on the FNs and paving the way for other multi-user operations such as shared downlink transmission [88].

5.2. Infrastructure design

5.2.1. Hardware abstraction

The Edge–Fog–Cloud is comprised of numerous nodes. These come in various configurations and are produced by different manufacturers. Consequently, there is a large discrepancy in their underlying hardware architecture. Moreover, their capabilities and abundance of resources – such as processing power and storage capacity – differ from one another. Infrastructure virtualization allows fog services to reside inside a homogeneous environment, no matter the hardware or operating system (OS) configuration of the FN. Standardized APIs then allow the services to access specific FN components such as the GPU or memory storage. Further clustering together multiple FNs with abstracted hardware resources makes the aggregated FNs behave as a large pool of resources [89]. When a service demands an increase in computational capabilities, additional resources are allocate to it – a concept that is otherwise common for the cloud.

To allow adequate fog resource continuity, the FC architecture needs a common infrastructure virtualization framework. Although VMs are common for the cloud, they are not a suitable solution for the fog. The large amount of overhead associated with VMs, compared to the small resource demands of individual services, would result in a poorly utilized fog infrastructure. For optimal resource allocation and a high QoS, the virtualization framework should be tailor made for each specific domain. For example, fog for vehicles needs to cope with high user mobility and low latency requests, all while operating on constrained hardware. Therefore a specific framework for encapsulating, migrating, and orchestrating services can be built on top of a wireless communications physical layer for vehicle-to-everything (V2X) communications such as the IEEE 802.11p [90]. Unfortunately, there are drawbacks in terms of flexibility. However optimized it may be for specific services, such infrastructure offers little diversity. For example, when a road worker needs to mend a malfunctioning fusebox, their AI-supported repair and maintenance operations (RMO) application cannot offload computationally demanding tasks to the same fog infrastructure. Instead, a more flexible approach is needed.

The container orchestration framework needs to be entirely hardware-independent. While some applications have very specific requirements, FC is intended for broad use. Two existing widespread frameworks already manage to achieve this goal: Docker Swarm and Google Kubernetes. Their main benefit is that they are already well established in the cloud. Consequently, they allow the fog to bridge the gap to the cloud and provide an endless continuum of resources. Regardless, they are not entirely suitable for the dynamic nature of constrained fog resources on their own. Swarm and Kubernetes were design with the stability and scalability of cloud resources in mind. That is why IoT and FC architects are attempting to adapt them for operation on fog infrastructure and devise their FC counterparts. Table 3 provides an overview of the more promising efforts.

5.2.2. Capacity and resource planning

Determining the optimal distribution of edge–fog–cloud resources is a tedious task of balancing out the location-based user requests with performance of the available hardware. It is an important aspect of architecture design, as an optimal solution will also minimize the cost and carbon footprint of infrastructure equipment while meeting the QoS requirements of edge devices. Most notably, latency constraints. Capacity planing is best done using custom or publicly available simulation frameworks. For instance, the framework presented in [73] determines the optimal combination of hardware components given a certain project budget. Due to its analogy to the optimal service placement problem, capacity planning can be handled by formulating and solving an integer programming problem [98]. Given capacity planning is carried out beforehand, there is no need to reduce the time and calculating and approximate solution as is the case for SP. Consequently, precise solvers, such as Gurobi or CPLEX are deemed appropriate for finding an optimal solution. To further illustrate, a common capacity planning procedure carried out by a simulation framework goes as follows [99]:

(1) Identify the services and the QoS requirements of tasks offloaded to these services,
(2) Profile and analyze the usage of system resources while the services are run,
(3) Separate tasks into those that must be processed in the fog and those that can be forwarded to the cloud,
(4) Determine which tasks have complementary resource demands to optimally use all of the FN's resources,
(5) Estimate the amount of needed edge nodes and their resources based on the amount of users in a geographic area.

Alternatively, taking into account potential uneven task offloading requests, the capacity planning problem can be formulated as a continuous-time Markov chain [100]. Such fog resource planning better compensates the stochastic nature of offloading requests. It also results in a fog configuration which will be more resilient during future changes in usage trends.

5.3. Provisioning, allocation, and offloading

Resource Provisioning refers to the providing of resources like memory and computing power of compute nodes used by IoT applications. Resource Allocation refers to the assignment of resources needed to handle an IoT request [19]. Resource allocation and resource provisioning go hand-in-hand because nodes must provide resources to allocate IoT requests. Allowing for continuously high QoS levels recorded by the edge devices, the ideal SP strategies adapt in real-time to changes in edge device demands, network infrastructure availability, and user mobility. The SP decisions can originate from various parts of the edge–fog–cloud model and, depending one the desired resilience, orchestration speed, and autonomy level. The following subsections compare the possible combinations of orchestrator location and SP strategies, while they also introduce the main constraints used for formulating SP problems.

5.3.1. Resource provisioning

- *Prior Provisioning*: In order to achieve prompt resource provisioning and allocation, the fog can save resources for future IoT use by predicting IoT traffic. When an IoT request is received, the fog assumes needed resources are available and processing occurs immediately. Any scheme that uses this approach is known as a prior provisioning and prompt allocation scheme [19].
- *On-demand Provisioning*: Because predicting IoT traffic requires extra computing and may not be accurate, the majority of schemes wait to provision resources when an IoT request is received. Resources can be assigned via on-demand provisioning using small batch allocation or prompt allocation.
  - *Prompt Allocation*: This approach requires that the request is transmitted to the closest fog node to be processed without consideration of cost. Prompt allocation is best for providing responsive IoT support [19].

Table 3
Container orchestration frameworks.

Name	Summary	Ref
FITOR	Modified Calvin IoT framework [91] for dispersed IoT application provisioning	[80]
EPOS Fog	I-EPOS [92] based framework for distributed resource orchestration	[72]
ETSI MANO mod.	ETSI MANO inspired framework, allowing autonomous fog CN resource orchestration	[93]
Docker Swarm mod.	Docker Swarm orchestration with added user-defined container deployment request scheduling	[94]
Google Kubernetes mod.	Google Kubernetes with modified scheduling, based on network infrastructure monitoring	[95]

*A broader comparison including other available orchestration frameworks can be found in [96] and [97].

– *Small Batch Allocation*: A fog resource manager can also gather small batches of IoT requests and identify the ideal allocation of IoT tasks that optimize a key metric like cost or latency. When a node is provisioned to handle an IoT request, any additional requests from the same application are promptly services by the node. Therefore, small bath allocation can efficiently handle IoT processing for the entire life of a provisioned module [19].

5.3.2. Service orchestration

- *Online vs. Offline:* Optimizing the SP problem beforehand enables the calculation of exact solutions and therefore yields the highest possible achievable QoS levels. Offline optimization is often tackled using off-the-shelf solvers, such as Gurobi, CPLEX, and Choco, and although a solver repeatedly provides promising results, it is extremely times consuming and often unsuitable for online orchestration. To cope with the constantly changing demands of edge devices and the corresponding fluctuation of FN resources, SP optimization needs to be repeatedly carried out in real-time. Reducing optimization times, heuristic and approximative algorithms with relaxed constraints are employed. They provide relatively consistent results and lower the optimization times from several minutes [101] to the order of seconds or even less, depending on the complexity of the problem [95,102].
- *Centralized vs. Hierarchical vs. Distributed:* A central orchestrator is positioned in the cloud and has a complete overview of both the fog and edge, enabling it to jointly optimize the SP problem and satisfy the global goals of the architecture. The orchestrator can rely on immense processing power; therefore, it is able to produce almost optimal SP solutions in real-time. Some examples include cloud-driven placement of VM replicas [75,84] and fog layer microservice placement coordinated by the cloud [79]. Granting an adequate amount of independence to the FNs, while still overlooking their coordination from a higher perspective, gives birth to hierarchical control schemes. Since FNs closer to the edge (consumer routers and gateways) have very limited resources, it is sometimes unwise to burden them with the task of full service orchestration. Instead, they may conduct simple local orchestration, while they themselves are controlled by a more powerful FN, residing one layer above them. The pattern continues, till the central orchestrator in the cloud is reached. This way the whole architecture is coordinated in timely fashion, while offering adequate resilience, as in the event of a failure the remaining FNs can continue to serve nearby edge devices interruptedly [77]. The architecture's resilience is further enhanced by allowing FNs on the same layer to directly communicate with one another [76]. The SP strategy of a FN is not always governed by a multi-node orchestrator, and the FNs can either cooperatively or individually optimize the SP problem. In non-cooperative fashion, each FN may adopt a cache-like strategy where it decides to migrate services strictly based on the demand of incoming offloaded tasks [103,104]. Although the so called *on-path* provisioning offers a relatively straight forward solution to distributed infrastructure orchestration, it demands high demands a non-negligible amount of attention from the FN and is only suitable for resource-rich FNs. On the other hand, FNs may also opt for coordinated distributed orchestration. Each FN communicates with other FNs in its vicinity through a direct secure channel while discovering nearby resources. The gathered information is then combined with the history of requests coming in from the edge devices, and an optimal SP solution is established. The solutions of individual FNs are then equally weighted and merged together to form the global optimum solution for service placement [72]. Such orchestration alleviates the damage inflicted by potential single-point failures; however, standardizing the inter-fog communication channel presents a challenge on its own. In addition to the above SP control strategies, the SP strategy may be imposed by the edge devices themselves. Although resource allocation remains a task that is handled by the FNs, the connected edge devices are responsible for triggering the migration of containerized services to the corresponding FN [105]. Compared to distributed orchestration, edge device driven service placement and migration offers superior levels of resilience. On the down side, it is not self-complete, and the FNs need to encompass additional load balancing mechanisms to avoid overloading.
- *Static vs. Dynamic:* To compensate for changes in the data transmission behavioral parameters (network traffic, wireless channel changes) and to support network resource discovery, the SP control logic needs to exhibit dynamic behavior. Dynamic SP is based on online service orchestration; yet merely re-running the SP optimization is not sufficient. The orchestration framework needs a way of observing the crucial parameters, such as E2E latency caused by network congestion [74], and dynamically adapting to the conditions — hence the name. Moreover, the SP control logic should also autonomously keep track of the inclusion of additional computing nodes into the network, without the need for manual intervention.

5.3.3. Service migration

For reference, some of the most common problem formulation techniques found in literature are listed in Table 4 together with the corresponding articles where they are addressed in detail. Multiple approaches to problem formulation and solving exist, depending on the various optimization metrics associated with the SP problems, the desired levels of precision, and the amount

Table 4
SP problem formulation and solving algorithms found in recent works.

Problem formulation	Solving algorithm	References
ILP	Approximation	[72,92,106]
	Heuristic	[75,79,81,107,108]
	Exact (CPLEX)	[77]
	Exact (Gurobi)	[74,79]
MILP	Heuristic	[57,78,109,110]
MIQP	Approximation	[71,76]
MDP	RL	[111–114]
	DRL	[115,116]

of time available for producing a result. The SP problem with its numerous optimization metrics is most often described as a form of integer problem. For instance, starting with a problem described using integer linear programming (ILP) with strictly discrete constraints, the goal is to first reduce its complexity so that it no longer takes the shape of a NP-complete problem. In order to aid the problem's solvability and to provide a solution in polynomial time, linear programming (LP) relaxation is commonly used. With the help of LP relaxation, the problem's integrality constraints are removed, transforming the original problem into a related LP problem solvable in polynomial time. Although the newly obtained problem yields merely an approximation of the initial problem's solution, the reduced computational time needed for finding a solution means it is more suitable for online orchestration. Other instances of problems described using integer programming include mixed integer linear programming (MILP) and mixed integer quadratic programming (MIQP).

Reinforce leaning (RL) represents another problem solving technique also suitable for online orchestration. Commonly applied to problems modeled as a Markov decision process (MDP), RL stands for a group of lightweight ML algorithms that excel at producing accurate results in timely manner. Typically, an optimization problem is modeled as a MDP when the outcome of each decision is partly under the control of the decision maker and partly random. For example, an edge device might estimate that offloading a task to the nearby FN will meet the task's latency constraints. However, other devices may take the same decision, causing a spike in processing demand at the FN. Consequently, the time needed to execute the initial edge device's task is influenced by the computational demands of the task itself, as well as other seemingly random offloading requests. Most often, solving SP problems modeled as a MDP is addressed using the widely adopted Q-learning RL algorithm. Although it does not provide the most accurate results at first, the Q-learning algorithm progressively improves its decisions based on the rewards and penalties associated with its past actions.

5.3.4. Constraints

This section provides a list of constraints that are typically incorporated in the service management systems:

- *Resource constraints:* To avoid unnecessarily commissioning redundant network infrastructure, the utilization of existing FNs needs to be maximized. Taking only latency into account, the SP optimization may force an individual FN into overload, while other FNs waste precious resources as they wait for a request to come in. Instead, the orchestration framework may take into account even resource utilization, giving more flexibility to the infrastructure in case of a sudden spike in demand for a specific service. For instance, by modeling the presence of a particular service instance at a FNs as a binary number $\mathcal{N}_i \in 0, 1$, the optimization algorithm should strive to equalize the total number of service instances N_{total} placed at individual FNs [72]. Although feasible in static scenarios, finding a solution to even resource utilization in mobile environments is a daunting task. Alternatively, edge devices can be grouped together based on their mobility within a particular geographical area, where they are then assigned a corresponding cluster of fog resources [117,118]. The more mobile a group of users is, the larger their clusters are in geographical sense, while the more numerous a group is, the more resources it is assigned.
- *Network Constraints:* Although both bandwidth and latency are the main factors distributed computing architectures, FC solves bandwidth bottlenecks by design. One of its goals is to reduce the amount of traffic whizzing through the network by intercepting and processing data as close to the edge devices as possible. Therefore, the remaining important network constraint is latency. Latency is one of the main motivations for moving services from the cloud to the fog and for offloading tasks from edge devices. It represents an edge device's total waiting time, from sending a request or offloading a task, to receiving a response. Latency is comprised of several components, represented by (1) the data rate of the RAT the edge device and AP are using, (2) the number of hops between the AP and given FN, and the request (3) queuing and (4) processing time at the FN. For E2E latency, components 1 and 2 apply twice. To achieve the lowest combined latency, edge devices in crowded areas often need to avoid using the nearest FNs because of long queuing times [78]; thus, striking a balance between individual latency component contributions is needed. In terms of communication latency between individual FNs, trying to balance their loads and forward edge device requests, a second latency variable – the inter-FN latency – is used [93]. Although the inter-FN latency does not directly affect the offloaded task, it determines how fast the distributed computing architecture's response time and indirectly affect the task processing delays. Therefore the distance between FNs and the means communication can play a pivotal role in latency reduction.
- *Energy constraints:* An immensely useful bi-product of FC in today's efforts towards a sustainable future is its potential to lower the total energy consumption of the network infrastructure. The previously debated battery life optimization problem can become an effort at reducing the entire architecture's energy consumption. As such, energy efficiency can be used alongside latency, resource utilization, and other parameters as one of the key metrics considered during the service placement problem formulation [119]. Since even FNs can rely on batteries as their main source of power, the optimization algorithm may consider over-utilization of FNs connected to mains power supply in favor of long-term operation of the entire fog layer [120].
- *Service Constraints:* Services may impose their own constraints, often associated with their structure or requirements, as listed below.
  - *Service Structure:* As discussed in Section 5.1, service may be monolithic, parallelizable, or modeled as a DAG. During SP optimization, the service structure is taken into account when aiming for an even utilization of fog resources. Generally, the larger the percentage of parallelizable and DAG services, the better the fog infrastructure is utilized.
  - *Locality:* Another constraint that may also provide a hindrance to equal service distribution are their locality requirements. For example, positioning services

need to run close to edge devices in order to provide adequate context awareness, while banking services are preferred to run within the premises of the bank for security reasons. Addressing these constraint during SP optimization is done by classifying services based on their locality requirements and imposing the newly defined classes as optimization constraints.

- *Latency:* The timeliness of execution is perhaps the most important constraint imposed by fog services and the corresponding offloaded edge device tasks. It is closely bounded to the aforementioned network latency and fog resources. The total task processing latency is defined as the local processing time and the processing time at the FN, plus the round trip time (RTT) of the wireless connection [121,122]. By also associating each service class with a latency constraint, the SP optimization algorithm can determine the optimal distribution of services, given a precise geographical distribution of user requests.
- *Edge device battery life:* From an individual edge device's point of view, CO may not only provide faster processing, but it may also extend its battery life. The edge device can include its own energy consumption as an appropriately weighted parameter in the computation offloading decision process and selfishly decide to offload tasks, centered on its own energy savings [123]. In considering the benefits of offloading, the edge device must take into account the tradeoff between the energy consumed for local computing versus the required power for (wireless) transmission and reception during CO.
- *Cost of provisioning:* Other optimization metrics can include the global financial aspects of placing services at different parts of the fog layer and the cost per single task processing at a particular FN. For instance, if an owner of a specific service wishes to distribute multiple such service instances across the fog, they can associate each FN in the SP optimization problem with a binary weight $\omega_i \in 0, 1$, promoting SP onto the fog infrastructure already owned by the service provider, instead of placing services onto billable 3rd party FNs [79]. Moreover, including additional billable infrastructure (mobile services) in the optimization problem allows the system to fully utilize all available resources, not just those associated with processing, and at the same time provide a better QoS-to-price ratio [124]. Edge devices may include the cost of offloaded task provisioning in their own CO decision making process similarly to how they determine the offloading decision based on battery life. The processing cost at an FN can be modeled using the price per unit of computing resources, multiplied by the size and complexity of the task [125]. The edge device then determines the time saving versus price ratio and includes it in the offloading decision.

5.4. Infrastructure and service monitoring

Since the SP optimization algorithms are intended to run in real-time, they require a set of advanced monitoring tools that grant them access to network and resource data, incorporated into the optimization problem. Fog and cloud computing both rely on resource and network utilization monitoring; yet, using the same monitoring platform without modifications for both of them is not always possible due to the additional constraints imposed by the fog. Whereas both cloud and fog require precise hardware resources and network bandwidth monitoring, FNs can only afford to run undemanding resource utilization collection tools, while FC in general needs to keep better track of additional network information such as data transfer latency [126]. Although the monitoring tools can reside in multiple layers, each FN should host its own set of monitoring tools. Whether or not it uses the data strictly for self orchestration or forwards it to a central control platform depends on the orchestration framework in use. While some authors rely on custom solutions [93,96,127], readily available monitoring tools are slowly starting to find their way into edge computing [80,94]. The more common examples include:

- *Resource utilization collection tool:* As in a live environment at any given moment numerous services might be getting migrated from one FN to the other or spiking edge device requests might be triggering the expansion of resources allocated to a particular service, each FN needs to keep precise track of their resources. The resource utilization tool can monitor only the basic three metrics, mentioned in previous parts of the present work – CPU usage, memory availability, and remaining storage [93,96] – or it can collect more detailed runtime data. For instance, cAdvisor [128] gives additional insight into how much resources are allocated to each container, making it an ideal tool for monitoring microservice based applications spread out across the fog.
- *Network probing tool:* To determine the possibility of offloading and forwarding tasks to other FNs, an FN must have a good overview of the network devices in its vicinity. Moreover, the FN needs to autonomously determine the state of the data link connecting it to other FNs, edge devices, and the cloud. As an example, Blackbox exporter [129] enables latency data collection between two network connected devices.
- *Central monitor:* Lastly, an FN needs to include a central monitor to interpret all of the gathered data. The central monitor interprets the observations and triggers additional events when needed. An event might be triggered because the FN is being overloaded and wishes to forward part of the requests to another FR, or simply to let the central orchestrator know that the bandwidth on a certain connection has reduced and subsequent requests need to be rerouted. Often used in custom fog orchestration frameworks [80,94], the Prometheus [130] monitor is a versatile open source tool, backed by a large community of developers and companies alike.

5.5. Performance evaluation platforms

When designing an optimization algorithm it is of paramount importance to select the appropriate modeling tools which closely mimic the environment in which the algorithm will be employed. Based on how closely a developer of SP and CO optimization algorithms wishes to mimic an everyday use case, and given the amount of flexibility they need in setting up the experiment, the evaluation platforms can be bundled into three groups:

(A) **Bare metal simulation:** Set up using the tools of choice, bare metal simulators offer the largest amount of flexibility. Whichever programming language, libraries, or OS the simulator is based on, with a tailor made simulation setup, the developer has an infinite degree of freedom in modeling the network infrastructure resources. Although resource modeling is commonly constrained to CPU performance, memory availability, and storage size; in a custom setup, the resource modeling parameters may include additional specific processing units such as GPUs or various

types of data storage devices. Furthermore, the developer can freely define the distances between FNs and the delays the corresponding communication links impose, while the arrival rate of offloaded tasks at FNs and their sizes can take the form of any given distribution model. In literature, custom simulators are most frequently designed in Matlab [110,113,118,131,132] and Python [79,84,133,134]. Given the distribution of the number of requests that arrive at an FN during a pre-defined time period, the incoming requests are typically modeled using a discrete Poisson distribution, where the λ parameter is fine-tuned to represent the average arrival rate of the request. While a Poisson distribution may be suitable for describing the incoming load on an FN per time unit, it cannot efficiently define the distribution of network propagation latency values. Instead, latency is modeled using the Gamma distribution, as its long-tail property accurately describes the increased network congestion and routing issues in wide area networks (WANs) [74]. A derivation of the Gamma distribution, the Erlang distribution, also proves useful for describing the average speed of individual users in highly-mobile environments [118]. Unfortunately, the high customizability of tailor made simulators makes comparing SP or CO algorithms presented in different works cumbersome, as the differences in how the simulators model resources and events introduces added uncertainty into the comparison.

(B) **Readily available simulation frameworks:** Among the more complex but relatively straight forward to use simulation platforms, iFogSim [135] stands out as the most widely adopted. iFogSim is based on the framework of CloudSim, and it is able to emulate all three device layers of the edge–fog–cloud model. Although it is not able to recreate the network infrastructure in all aspects, it does associate one of the following parameters with each fog or cloud device: CPU performance, memory availability, storage size, and available uplink and downlink bandwidth. With exposing the listed device parameters and relying on CloudSim at its core, iFogSim enables developers to simulate multi-user scenarios and observe QoS metrics while optimizing SP algorithms [77,81]. Furthermore, iFogSim serves as a basis for other simulators such as MyiFogSim [136], while the general inclusion of CloudSim in fog simulators [137,138] demonstrates the versatility of cloud modeling tools in FC. When developers want to use a ready-made simulator, but they wish to have better control over the simulator's behavior, they may decide to use another Java based framework — the open source FogTorchΠ simulator [124,139]. By default, the definitions associated with the network infrastructure allow for a similar degree of freedom in configuring the edge–fog–cloud architecture to that offered by iFogSim, while FogTorchΠ also considers the financial aspects of the edge–fog–cloud model [73].

(C) **Testbed:** Bridging the gap between a controlled environment and a real-life use case, testbed setups give developers an in depth overview of the behavior of the architecture in multi-user applications running on consumer electronics. Amid the off-the-shelf devices, individual desktop computers are normally used for simulating FNs [140], smartphones can be used to represent the average edge node [87, 132], while versatile single board computer (SBCs), such as Raspberry Pis, can be individually designated as edge nodes or clustered together to represent and FN [58]. The affordability of the latter makes SBCs especially useful for assembling large testbeds, comprised of tens or hundreds of different devices. Although the costs of constructing such a testbed add up, developers and researchers worldwide can leverage the infrastructure of existing testbeds setups to fine-tune their designs. Examples of large scale IoT and parallel computing experiment platforms include the well renowned FIT/IoT-LAB [141] and Grid5000 [142] testbeds.

5.6. Fog computing frameworks

Fog orchestration refers to the use of a control layer to track resources and request resource allocation for fog nodes. IoT requests are uploaded to the control layer first and then the application is sent to the cloud or fog nodes. A fog framework can be viewed as a set application of fog orchestration intended to support the application and provisioning and allocation of resources using Fog Orchestration Controller (FOC) [38]. Typically, FOC periodically queries the available resources, manage the request to fog and distributes the tasks among fog nodes. Note that depending on the number of layers and clusters of fog, there might be more than one FOC in the system. In that case, FOCs can communicate with each other to distribute the load. In this section, we briefly overview a few of existing FC frameworks [19]:

- **Fog-as-a-Service-Technology (FA2ST)**: This framework supports on-demand discovery of fog services, permitting the framework to check connected nodes for available resources when an IoT request is received. In addition, if a node is not available to handle an assigned IoT application, FA2ST can send the application to another node [143].
- **Storm**: The authors of [23] extended Apache Storm (an open-source distributed real-time computation system) to realize a distributed QoS-aware scheduler with self-adaptation capabilities for handling data stream applications. In this approach, each fog node is aware of the resource availability of other nodes. Thereby, it enables the IoT service scheduler to work within resource and latency requirements making it suitable to operate in a geographically distributed and highly variable environment.
- **Fog–IoT Orchestrator (FITOR)**: This framework monitors fog infrastructure to compute and extract a set of resource utilization metrics for every node. This gives FITOR a comprehensive view of all fog resources, enabling the deployment of IoT data services to fog nodes [144].
- **FogBus**: FogBus framework consists of gateway nodes, repository nodes, computing nodes, and broker nodes. IoT data or tasks is uploaded to a gateway node. In the next step, the load is propagated to broker nodes before being delivered to computing nodes to process the request. The broker nodes are connected to repository nodes that facilitate the sharing, recovery, storage, and replication of data [40].
- **SDN Controller**: SDNs separate the data and control planes in a conventional network, forwarding data throughout the network using an arbitrary set of rules. Adding an SDN to a standard switch (e.g., OpenFlow) creates a simplified interface for interactions between SDNs and the network. In addition, an SDN is capable of tracking mobile IoT devices, anticipate destinations, and facilitate seamless data transfers among fog nodes. The authors of [42] suggest managing resources, data movement, and controlling traffic in the fog plane using an SDN with OpenFlow. The Mist framework utilizes an SDN to monitor resources and employs APIs to keep an eye on fog and IoT device health and help with IoT service allocation [38].

5.7. Edge–fog–cloud computing enabled AI for IoT

5.7.1. Existing solutions and trends

The fast paced development of machine learning (ML) algorithms, combined with the advancing computational capabilities

of computer hardware has yielded a number of both proprietary and open-source ML frameworks, executable on diverse hardware [145]. However, executing a complex ML algorithm solely on an edge device is not always feasible due to the lack of computational resources. Contrarily, running the ML algorithm in the cloud does provide sufficient resources; yet, it imposes additional transport delays and can take up a large amount of bandwidth. Combining both approaches, and by distributing the ML algorithm between the edge and the cloud solves some of the aforementioned problems, while the benefits of distributed ML are even further enhanced when the entire edge–fog–cloud architecture is employed for data processing. For a better perspective on the latest developments in distributed ML, Table 5 outlines several examples found in recent studies.

5.7.2. Privacy-preserving, collaborative and distributed ML: The role of edge–fog–cloud

Managing the massive quantity of data created by IoT sensors requires an distributed edge–fog–cloud computing hierarchies. Because ML model training on edge devices is arduous mainly due to computing and memory limits, the majority of applications uses the cloud to train the models. Some have tried using model quantization and pruning to train models on edge devices, but this often results in reduced accuracy. In general, offloading tasks from a central cloud server to edge nodes results in a substantial reduction of transmitted data; however, ML models (e.g., DL networks) need a massive amount of computational resources including energy and memory. For example, Conventional Neural Networks (CNNs) used in computer vision tasks need substantial memory to store intermediate outputs, big datasets, and weight parameters. This requirement can create bottleneck issues when ML is used with edge/fog computing. Successfully using ML/DL when resources are limited in an edge framework requires that the DL network working on an edge server be furnished with many separate implementations represented as "service levels". Each distinct implementation of the network generates the same classification or prediction with varying accuracy versus resource in a way that a higher service level will produce greater accuracy. The idea of "service levels" and task offloading should be combined to distribute the intelligence across edge–fog–cloud to address power/energy consumption, latency, and performance constraints of the target IoT application [8].

Attempting to resolve issues around access to massive amounts of IoT data, researchers have focused on collaborative ML. This type of ML permits multiple participants, known as distributed trainers, to train a ML model in partnership. However, collaborative learning results in privacy concerns. To tackle this challenge, the concept of privacy-preserving ML (PPML) and Multi-Party Computation (MPC) have been developed enabling stakeholders to conduct analysis on private data without ever revealing those inputs. The stakeholders get no further information about each other's data, except from what can be learned from the result — public to all involved stakeholders [156]. Many IoT applications across vertical domains can benefit from PPML and MPC. For instance, in a manufacturing environment, the manufacturers of machinery rely on Machine Generated data (MGD) to offer predictive maintenance capabilities. However, ML algorithms need data containing privacy-sensitive information about the machine owners and their business. This burning issue is addressed by PPML and MPC, which enable analyzing the combined privacy-sensitive data from many machines. Below is a brief summary of four techniques that aim to address the challenges concerning ML and privacy [156,157]:

- **Federated Learning**: A decentralized ML technique, Federated Learning (FL) typically combines the computational capabilities of one main central server and several client servers; it ensures that, throughout the learning process, all private data remains at the location of its origin. In FL, the central server trains an initial model on proxy data available beforehand. The initial model is then sent to the clients for local training using their own local data, where the model replicas are independently trained for a number of iterations. This means that each separate contributor gets a copy of the model's parameters from a central server, trains it with some private data it owns, and sends back the computed gradients/coefficients of the model to the central server. The updates received from multiple clients are then combined in the central server to generate one global model. Finally, the server sends the improved model back to the client servers participating in the training. The above steps are iteratively executed, as the learning process is continuously repeated until a desired level of accuracy is achieved [156].
- **Differential Privacy**: In the Differential Privacy (DP), data sources such as edge devices relay a slightly altered version of the raw data to a central server. The basic idea behind this technique is that each contributor adds noise (i.e., random number) to its raw data before sending the data to the central server, hiding privacy-sensitive data from eavesdroppers. The collected noisy raw data are then combined in the central server, where they are used as input data for any given ML model. Hence, the data aggregated and processed in the central server is kept safe and hidden from 3rd parties at all times.
- **Homomorphic Encryption**: The centralized ML technique, homomorphic encryption (HE), is a kind of encoding that allows certain operations to be performed on the encrypted data, while preserving the initial encryption keys. When decrypted, it contains the same result that would have been obtained if equal operations were performed on the unencrypted raw data. Therefore, in HE, a contributor sends encrypted data to the server, where it gets analyzed without the server needing to first decrypt it.
- **Secret Sharing**: In the secret sharing technique, privacy-sensitive data from each participants/party is distributed among other parties in the form of *shares/secret*. If these shares/secrets are misused by a malicious (third) party, they would be worthless because they are decipherable only once they are combined together. Generally, this technique relies on two phases. The first one, usually called the offline phase, consists of a trustworthy *dealer* separating the data into shares and distributing them among the contributors. After that, the second phase, usually called the online phase, is where the actual computations are carried out. Once processed, the data can only be reconstructed once a sufficient amount of contributors combine their secret data and obtain the necessary key for deciphering it.

5.8. Security and privacy perspective

In edge–fog–cloud IoT architecture, security and privacy should be addressed in an end-to-end approach, covering all components between the endpoint device and the cloud. Deployments require the implementation of security mechanisms designed around a threat model and the asset's value. Privacy is a data property defined as the ability to choose how information is utilized. IoT architectures must enable user to choose data privacy characteristics for data they own within the system. Systems with multiple users may have to choose privacy attributes as well as data sharing rights between users. Edge–Fog–Cloud IoT systems that collect data to analyze it must also consider data privacy in the

Table 5
An overview of present distributed machine learning applications and future trends. The use cases are separated into the largest group, neural networks, and the remaining ML models.

	Use case
Neural network	Sequential data processing using neural networks (NN), where the privacy-sensitive data is pre-processed on the edge device and the remaining tasks are offloaded to a nearby computational node [146]. Distributed image processing based on a convolutional neural network (CNN), spread across multiple network layers for power consumption and data transfer amount reduction [147,148]. Similar to [147,148], whereas high processing power of the cloud is leverage for training the feed-forward neural network (FFNN), which is then transferred back to the fog layer [149]. With an alternative approach to CNN image classification, in [150] the authors distribute the ML model layers among the fog and the cloud; however, if the layers in the fog successfully classify the images, an early-exit strategy prevents unnecessary uplink between the fog and the cloud. In [151] the authors address the problem of distributing CNN layers between the edge device and corresponding computing node in real-time using an additional regression ML model, primarily based on the observed network latency.
Other	Two-stage linear regression (LR) model, with one part residing in the edge device and the other in the corresponding FN. By simulating sensor data at both stages, and only reporting trend changes to the higher hierarchical layer, the uplink data from the edge to the cloud is significantly reduced [152]. Distributed EEG data classification by combining logistic regression on a low-power IoT device and gradient boosting on a nearby smartphone [153]. To control electric loads in smart cities, lightweight DRL-based electric grid controller replicas are migrated to FNs where they can take swift action in balancing electric loads, while a central DRL in the cloud further orchestrates the grid [154]. [15]

*An in-depth comparison between distributed and single-device ML algorithms can be found in [155].

deployment. Creating secure IoT solutions that extend end-to-end demands a comprehensive approach that includes many levels and combines security features spanning the Communications, Lifecycle Management, Device, and Cloud layers, as illustrated below:

- Device Layer: The attacks most often experienced at the device layer include spoofing, tag cloning, direct connection, RF jamming, and cloud polling. Cloud polling attacks attempt to reroute network traffic in an effort to add malicious commands to a device. This can be done through a Man-in-the-Middle (MITM) attack and changes to Domain Name System (DNS) settings. During a direct connection attack, the Service Discovery Protocol (e.g., SSDP/UPNP) can be exploited to find and discover IoT device. Resolving such attacks requires that unauthentic requests be blocked by IoT devices through strong key management and cryptographic algorithms. Typically, device layer security measures include authorization management, identity, secure booting, authentication, whitelisting, data protection during capture, storage and transmission, application sandboxing, tight access control of resources, fault tolerance, password enforcement, secure transmission mechanisms, traffic filtering, and secure pairing protocols. When implementing security algorithms on IoT devices, limited resources such as power, range, memory, processing, and embedded network protocol/OS must be considered. The Key IoT security features of this layer include [1,158]:
  - Chip Security: Some producers are including chip security through Trusted Platform Modules (TPMs) that function as a trust foundation by guarding credentials and sensitive data.
  - Secure Booting: This booting is utilized to guarantee that only verified software will work on a device.
  - Physical Security: The most common technique at this layer is anti-tamper mechanisms that fall into four categories including, Evidence, Resistance, Response, and Detection. Physical security must be supported by the fog node and align with the device's threat model and security policy. This determines the difficulty involved with accessing components and the subsequent consequences of an intrusion. The node's location and physical access level in the location are important to the security evaluation. Fog nodes in public areas like a utility pole or shopping center are more vulnerable to a physical attack.
- Communications Layer: This layer is responsible for IoT connectivity networks and unsecure channels are vulnerable to attacks. The most common attacks on this layer include MITM, sinkholes, eavesdropping, sleep deprivation, and Sybil attacks. Securing the network layer requires using trustworthy point-to-point encryption, message integrity verification (e.g., MD5 or SHA hashing), and routing mechanisms. The cryptographic algorithms used for encryption fall into two categories: symmetric algorithms (e.g., Skipjack, AES, Blowfish, DES) or asymmetric algorithms (e.g., NtruEncrypt, Elliptic Curve, Rabins Scheme). Symmetric algorithms are not as computation heavy as asymmetric; therefore, such algorithms are better for use with lower power 8 or 16-bit IoT devices. Vital IoT security features of this layer include [1,158]:
  - Data-Centered Solutions: These security solutions make sure data is appropriately encrypted while transported or resting so that if intercepted, it has meaning only for approved systems, applications, people, or devices.
  - Firewalls & Intrusion Prevention System: These solutions monitor data traffic flows ending at the device to discover intrusions and protect the communication layer from attack.
- Cloud Layer: The cloud most often falls victim to SQL or malicious code injections, DoS attacks, Spear-Phishing, path traversal, sniffing, cross-site scripting, execution of remote code, brute force attack, viruses, and Trojan horses. This layer includes key IoT security aspects including [1,158]:
  - Sensitive Information Storage: Data at rest in the cloud should be encrypted in order to avoid exposure to

attacks if a compromised third party with less security accesses data for something like advanced analytics.
  - Third-Party Verification: Confirming the integrity of third-party applications or cloud platforms attempting to communicate with cloud services provides protection against malicious acts.
  - Digital Certificates: The certificates are vital for meeting authentication and identification needs at the IoT scale.

- Human Layer: IoT security requires that people be trained in how to appropriately handle data, especially eHealth medical data. This is especially important because attackers that gain physical access to an IoT device are able to read internal firmware/memory and reconfigure settings to gain partial or complete device control. Users of IoT eHealth devices should avoid using weak passwords, sharing electronic or physical keys, and buying used medical equipment [1,158].
- Lifecycle Management Layer: This refers to the layer housing over-arching processes that maintain and update security for IoT solutions ensuring appropriate security from manufacture to installation and device disposal. Vital features include [1,158]:
  - Activity Monitoring: This is key to tracking, detecting, and documenting any suspicious acts.
  - Security Patches: IoT applications and devices require consistent patching to remain current, resist attack, and address any vulnerabilities.
  - Secure Remote Control: This is vital in order to maintain millions or billions of IoT devices.

6. Open issues and challenges

The edge–fog–cloud computing model heralds unprecedented improvements in IoT solutions, and it is set to forever change the way IoT architectures are designed. However, adapting the existing network architecture for multi-layer computing possesses a great deal of challenges. We have in the previous sections discussed several key enablers for achieving joint edge–fog–cloud computing. Most of the studies addressing them focus on advancing the state-of-the-art in a particular field. In the current section, we provide an overview of the challenges that lay ahead by organizing the open research questions and the corresponding research opportunities and future directions into 15 distinct groups and briefly summarizing each of them. The contents are based on the literature assessed in the present work.

(1) **Task Offloading**: The ability for edge devices to rely on resource rich remote computing nodes for carrying out computationally demanding tasks fosters the offloading of tasks such as real-time sensor data analysis and decision making. In order not to waste precious resources and time, an edge device needs to rely on offloading decision logic that balances out the trade-offs of local and offloaded computing and provide a trustworthy conclusion in the blink of an eye. Despite improvements in Offloading techniques, e.g., Device to Edge, Edge to Fog, Fog to Fog, Cloud offloading to edge/fog, there are still many challenges that need to be tackled. For instance, monitoring, service quality, delivery, robustness, fault analysis, latency, energy consumption, load balancing, security, privacy, and integration.

(2) **Service Placement**: Further reducing the data transfer latency and network load by distributing services across multiple network layers, carefully planned SP is hailed as a key enabler of combined edge–fog–cloud computing. However, continuous optimal SP needs to take into account computational node resource availability, task offloading demands, data transfer propagation delays, and user mobility — all in real-time.

(3) **Load Balancing**: Since task offloading and service placement are two independent mechanisms, an increase in demand and failure to respond may result in a computational node being overwhelmed with processing requests. In such cases, the load needs to be redistributed among the computational node's less burdened peers, maintaining high quality of service QoS levels. The associated challenges include resource discovery, problem modeling, and decision making — the latter being similar to the task offloading dilemma of edge devices.

(4) **Heterogeneous Resources**: The abundant multi-vendor fog infrastructure will be implemented on various hardware and OS configurations, entailing the need for infrastructure virtualization. Although CC offers some solutions, they are not fully suitable for FC. The latter will need a containerization framework with minimal overhead, capable of abstracting the diverse hardware found in the fog through a unified API.

(5) **Resource Management**: The goal of FC to provide computing capabilities to a broad range of services means the scale of available versus busy fog resources is bound to teeter over time. The orchestration framework will therefore also need to match and execute complementary services at individual FNs. Service matching is a common process during capacity planning [99]; yet, it will only truly even out the utilization of fog resources when repeatedly executed in real-time.

(6) **Failure Management**: Aggregating abstracted fog infrastructure allows services to demand more resources during runtime, similarly as it is done in CC. Unlike the cloud, the fog is not comprised of stable resources. FNs may enter and leave their clusters at any time due to the installment of new devices, small-scale network errors, or other local activities. This leaves it to the orchestration framework to provide additional safety mechanisms to services requiring high levels of resilience. For example, industry 4.0 services coordinating the movements of a group of robot arms. The orchestrator can provide adequate resilience to the service by placing multiple synchronized copies of the service at different FNs, providing redundancy for combating single points of failure.

(7) **Inter-Fog Communication**: Residing in multiple layers of the network, an FN can communicate in two directions: vertical (north/south-bound) and horizontal (east/west-bound). The common tree network topology permits the first, while horizontal communication has to pass through at least one additional layer. The additional network hops add to the communication delay and increase the fog's susceptibility to failure. To enable fast and reliable horizontal communication, additional connections between fog nodes residing on the same level need to be established. This can take the form of a direct physical connection between the two or it can be done via a wireless device-to-device (D2D) data link. Establishing wireless links within a fog layer, such as two APs, also means the FNs need to include a built-in service for D2D communication. Although the latter is an existing concept, FN D2D communication will be a topic of future FC standardization efforts.

(8) **Support for Mobile Users**: Such as tailor made standards for communication between highly mobile users exist,

there have also been efforts at designing fog architectures capable of processing their requests [90]. Unfortunately, these address only a specific group of the FC target audience. Future fog roll-outs will have to include multiple user types, including high-mobility users. One top-down approach is placing each edge device into separate user groups, one of which reflects their mobility pattern, and distributing fog resources according to their numerosity [118]. Such approaches solve part of the problem, and they could be further enhanced using advanced preemption-based scheduling schemes.

(9) **Hardware Technologies for the Fog**: The fog is envisioned to stretch a multitude of devices between the network's edge and its core. A representative example are APs and other consumer-grade gateways. At the time being, most of them are based on application specific integrated circuits (ASICs), with little or no flexible computing resources. Thus, these devices act merely as data forwarders and cannot participate in FC. Although one can buy purpose-built IoT gateways, these tend to be expensive and are not available in most consumer electronics stores. On the contrary, fitting each AP with additional CPUs and GPUs could considerably increase its price, putting into question the sensibility of the investment. There is no definitive answer; however, making consumer-grade equipment more flexible – by substituting part of the ASIC with a versatile CPU, or allowing the acquisition of CPU and GPU extensions – would help spur the widespread adoption of FC.

(10) **Low Latency Throughput Wireless Communication**: Transporting data from edge devices to nearby FNs and bridging the physical gap between the edge and the fog presents its own set of challenges. For one, the sheer amount of data transfer required by applications such a AR can result in bandwidths exceeding 1 Gbps [159], rendering many widespread wireless communication technologies useless. Moreover, the contribution to data transfer latency of communication technologies commonly found in IoT devices is in the range of tens of milliseconds [160,161], making task offloading less rewarding for edge devices. Lately, the millimeter wave (30–300 GHz) spectrum has been the subject of accelerated research, and as the underlying technological standards mature, more edge devices are starting to support multi-gigabit wireless communication [162,163]. Furthermore, by pushing the envelope of data transfer latency below 1 ms [164,165], the ultra-reliable low-latency communication (URLLC) features of 5G new radio (5G NR) are the harbinger of future real-time wireless communication.

(11) **Financial Model and Pricing**: The FC paradigm brings with it a host of new devices to the network architecture. Because an FN's owner may be a physical person, a healthcare institution, or a company providing other network services, the task of including them in a conventional centralized pricing system is far from trivial. Considering the geographical dispersion of FNs, the mobility of the edge nodes, and potential flexibility in the QoS demands of an edge device, a dynamic financial model is more suitable for the edge–fog–cloud computing architecture. In a simple dynamic pricing model, a computational node, that has processed an offloaded task, bills the edge device with an amount corresponding to the processing time duration, multiplied by a fixed cost per time processing unit [166]. Depending on how far from the edge device the computational node is positioned, it may request an additional *QoS improvement tax*, proportionate to the latency reduction caused by shorter network propagation times [104]. Given a request has a certain budget, it may be admitted at an FN or get propagated further through the network, till it reaches the cloud and the *QoS improvement tax* reduces to zero. The pricing models can be also in favor of the edge devices, as some authors propose that idle edge devices contribute their processing power to the fog and make profit, while helping the FNs meet increasing task offloading demands [64]. Whichever the case, there is still room for improvement, as an offloaded task may request more than just CPU-based processing, and a strictly profit-driven model may provoke continuous preemption of tasks with a lower budget, leading to their starvation [134]. Furthermore, a mechanism for keeping track of an edge device's consumption, and a standardized payment method will be required for offloaded task provisioning. While blockchain may provide a means of how edge devices can issue payments directly to the computational nodes [167], it is still not clear what the long-term solution will be.

(12) **Data Analytics**: Creating efficient algorithms to train ML models (e.g., neural networks) on edge devices is a growing area of research interest. It is important to be able to share data and computation outputs across diverse edge devices when developing an effective distributed system. New computation-aware network models have become popular for creating distributed data-sharing systems. 5G networks used to supply highly-reliable, low-latency communication are a primary target for integration with edge computing. In fact, 5G should allow greater control of network resources to support on-demand interconnection of diverse edge devices. In addition, adapting software-defined networks and network virtualization to become 5G networks that can control ML settings will offer new research opportunities.

(13) **Standardization and Interoperability**: The fog is slowly growing out of its nascent state, and it should not be long before FC becomes a part of everyday life and users worldwide get to experience its benefits – a pivoting point of which is rigorous standardization. The OpenFog consortium – founded in 2015 by Ciso, Intel, Arm, and others – dove deep into the FC paradigm, and proposed a reference architecture for FC, which later took the form of the 1934–2018 IEEE standard [168]. At roughly the same time, NIST proposed their own fog conceptual model [169]. The latter is slightly more compact than the OpenFog architecture, while it does additionally cover the mist layer. Several other distributed computing models have also recently undergone thorough standardization, most notably 5G multi-access computing, and although the individual architectures and concepts are being meticulously perfected, their interaction and cooperation remain to be defined.

(14) **Quality of Service**: In the context of edge–fog–cloud IoT, the service quality is very hard to guarantee mainly due to the inherent heterogeneous, dynamic, and distributed nature of the system as well as communication, I/O or storage constraints, and compute power. Beyond doubt that resource availability, service response time, fault-tolerance, and r efficient allocation/placement of tasks in a combined edge–fog–cloud paradigm are among important aspects that need of further research to meet the challenges for granting QoS [31,170–172].

(15) **Security and Data Privacy**: Security and privacy (confidentiality, integrity, availability, accountability, and privacy) remain as one of the most challenging and important issues to the popularity of edge–fog–cloud. The current research and progress on security and privacy protection

is still in the infant stage and lacks a unified methodology. The inherent uncontrolled, out-of-surveillance operation, virtualization technology, and related multi-tenant model, lack of standardized evaluation criteria, lack of fine granularity and elasticity of access control, lack of considerable progress in the construction of revocable, traceable, expressible, and efficient attribute encryption, lack of scalable edge-driven privacy-preserving decentralized ML techniques, new requirements for lightweight data encryption methods and fine-grained data sharing systems, resource-constrained edge devices, and lack of multi-sources heterogeneous data dissemination control are some important challenges that should be solved urgently [31,173–177].

7. Conclusion

IoT revolution is characterized by the convergence of technologies – from cloud computing to edge/fog computing, AI, ML, and big data – that is blurring the lines between physical and digital worlds. In particular, Cloud computing has exactly what is needed to fix the shortcomings of IoT by providing virtually unlimited computing and storage resources for better handling the massive deluge of data generated by IoT devices. In addition, fog and edge computing paradigms with the capability to move the computation closer to endpoint IoT devices have emerged to address the issues of energy efficiency as well as latency and context-awareness requirements. Combining the capabilities of edge, fog, and cloud creates a hierarchical IoT paradigm (i.e., Edge–fog–Cloud) to enhance IoT system functioning. However, the adoption of the hierarchical edge–fog–cloud is still facing various challenges, e.g., standardization, the interplay of edge–fog–cloud, and task offloading, among others. This paper geared towards a holistic reference covering all important aspects of the convergence of edge–fog–cloud and IoT to foster the research in this multidisciplinary field. In this paper, we first presented the state-of-the-art edge and fog computing models, explained their building blocks and components. Next, we demonstrated a holistic reference architecture for edge–fog–cloud IoT model, its features, benefits, and the interplay among the corresponding components. We also provided a tutorial on the related computing paradigms, including their technical similarities and differences. Following this, we discussed the major design, implementation, and deployment topics, such as infrastructure design, offloading, provisioning, placement, security, and privacy, as well as distributed and collaborative analytics. Finally, we discussed the key open issues and challenges.

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.

References

[1] F. Firouzi, K. Chakrabarty, S. Nassif, Intelligent Internet of Things: From Device to Fog and Cloud, Springer, 2020.
[2] B. Farahani, F. Firouzi, M. Luecking, The convergence of IoT and distributed ledger technologies (DLT): Opportunities, challenges, and solutions, J. Netw. Comput. Appl. 177 (2021) 102936.
[3] F. Firouzi, B. Farahani, Architecting IoT cloud, in: Intelligent Internet of Things, Springer, 2020, pp. 173–241.
[4] F. Firouzi, B. Farahani, M. Weinberger, G. DePace, F.S. Aliee, IoT fundamentals: Definitions, architectures, challenges, and promises, in: Intelligent Internet of Things, Springer, 2020, pp. 3–50.
[5] F. Firouzi, B. Farahani, M. Ibrahim, K. Chakrabarty, Keynote paper: from EDA to IoT ehealth: promises, challenges, and solutions, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 37 (12) (2018) 2965–2978.
[6] A. Botta, W. De Donato, V. Persico, A. Pescapé, Integration of cloud computing and internet of things: a survey, Future Gener. Comput. Syst. 56 (2016) 684–700.
[7] 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) 1–36.
[8] B. Farahani, M. Barzegari, F.S. Aliee, K.A. Shaik, Towards collaborative intelligent IoT eHealth: From device to fog, and cloud, Microprocess. Microsyst. 72 (2020) 102938.
[9] B. Farahani, M. Barzegari, F.S. Aliee, Towards collaborative machine learning driven healthcare internet of things, in: Proceedings of the International Conference on Omni-Layer Intelligent Systems, 2019, pp. 134–140.
[10] N. Mohan, J. Kangasharju, Edge-Fog cloud: A distributed cloud for Internet of Things computations, in: 2016 Cloudification of the Internet of Things, CIoT, IEEE, 2016, pp. 1–6.
[11] Z. Nezami, K. Zamanifar, K. Djemame, E. Pournaras, Decentralized edge-to-cloud load-balancing: Service placement for the internet of things, 2020, arXiv preprint arXiv:2005.00270.
[12] H. Shah-Mansouri, V.W. Wong, Hierarchical fog-cloud computing for IoT systems: A computation offloading game, IEEE Internet Things J. 5 (4) (2018) 3246–3257.
[13] J. Kang, D.-S. Eom, Offloading and transmission strategies for iot edge devices and networks, Sensors 19 (4) (2019) 835.
[14] S. Svorobej, P. Takako Endo, M. Bendechache, C. Filelis-Papadopoulos, K.M. Giannoutakis, G.A. Gravvanis, D. Tzovaras, J. Byrne, T. Lynn, Simulating fog and edge computing scenarios: An overview and research challenges, Future Internet 11 (3) (2019) 55.
[15] P. Hu, S. Dhelim, H. Ning, T. Qiu, Survey on fog computing: architecture, key technologies, applications and open issues, J. Netw. Comput. Appl. 98 (2017) 27–42.
[16] 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.
[17] 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) (2017) 416–464.
[18] P. Borylo, A. Lason, J. Rzasa, A. Szymanski, A. Jajszczyk, Energy-aware fog and cloud interplay supported by wide area software defined networking, in: 2016 IEEE International Conference on Communications, ICC, IEEE, 2016, pp. 1–7.
[19] I. Martinez, A.S. Hafid, A. Jarray, Design, resource management and evaluation of fog computing systems: A survey, IEEE Internet Things J. (2020).
[20] 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.
[21] P. Bellavista, J. Berrocal, A. Corradi, S.K. Das, L. Foschini, A. Zanni, A survey on fog computing for the Internet of Things, Pervasive Mob. Comput. 52 (2019) 71–99.
[22] M.I. Bala, M.A. Chishti, Survey of applications, challenges and opportunities in fog computing, Int. J. Pervasive Comput. Commun. 15 (2) (2019) 80–96.
[23] V. Cardellini, V. Grassi, F.L. Presti, M. Nardelli, On QoS-aware scheduling of data stream applications over fog computing infrastructures, in: 2015 IEEE Symposium on Computers and Communication, ISCC, IEEE, 2015, pp. 271–276.
[24] M. Mukherjee, L. Shu, D. Wang, Survey of fog computing: Fundamental, network applications, and research challenges, IEEE Commun. Surv. Tutor. 20 (3) (2018) 1826–1857.
[25] 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) 1–43.
[26] J. Ni, K. Zhang, X. Lin, X.S. Shen, Securing fog computing for internet of things applications: Challenges and solutions, IEEE Commun. Surv. Tutor. 20 (1) (2017) 601–628.
[27] M. Aazam, S. Zeadally, K.A. Harras, Offloading in fog computing for IoT: Review, enabling technologies, and research opportunities, Future Gener. Comput. Syst. 87 (2018) 278–289.
[28] A. Aljumah, T.A. Ahanger, Fog computing and security issues: A review, in: 2018 7th International Conference on Computers Communications and Control, ICCCC, IEEE, 2018, pp. 237–239.
[29] F.A. Salaht, F. Desprez, A. Lebre, An Overview of Service Placement Problem in Fog and Edge Computing, Tech. Rep. RR-9295, Lyon, France, 2019.
[30] D.S. Linthicum, Connecting fog and cloud computing, IEEE Cloud Comput. 4 (2) (2017) 18–20.
[31] D. Linthicum, Responsive data architecture for the Internet of Things, Computer 49 (10) (2016) 72–75.

[32] X. Masip-Bruin, E. Marín-Tordera, G. Tashakor, A. Jukan, G.-J. Ren, Foggy clouds and cloudy fogs: a real need for coordinated management of fog-to-cloud computing systems, IEEE Wirel. Commun. 23 (5) (2016) 120–128.
[33] V. Stantchev, A. Barnawi, S. Ghulam, J. Schubert, G. Tamm, Smart items, fog and cloud computing as enablers of servitization in healthcare, Sensors Transducers 185 (2) (2015) 121.
[34] OpenFog Consortium, OpenFog reference architecture for fog computing, 2017.
[35] B. Farahani, F. Firouzi, K. Chakrabarty, Healthcare iot, in: Intelligent Internet of Things, Springer, 2020, pp. 515–545.
[36] B. Farahani, F. Firouzi, V. Chang, M. Badaroglu, N. Constant, K. Mankodiya, Towards fog-driven IoT eHealth: Promises and challenges of IoT in medicine and healthcare, Future Gener. Comput. Syst. 78 (2018) 659–676.
[37] B. Tang, Z. Chen, G. Hefferman, T. Wei, H. He, Q. Yang, A hierarchical distributed fog computing architecture for big data analysis in smart cities, in: Proceedings of the ASE BigData & SocialInformatics 2015, 2015, pp. 1–6.
[38] K. Intharawijitr, K. Iida, H. Koga, Analysis of fog model considering computing and communication latency in 5G cellular networks, in: 2016 IEEE International Conference on Pervasive Computing and Communication Workshops, PerCom Workshops, IEEE, 2016, pp. 1–4.
[39] M.I. Naas, P.R. Parvedy, J. Boukhobza, L. Lemarchand, iFogStor: an IoT data placement strategy for fog infrastructure, in: 2017 IEEE 1st International Conference on Fog and Edge Computing, ICFEC, IEEE, 2017, pp. 97–104.
[40] 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.
[41] 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, 2016, pp. 258–269.
[42] S. Tomovic, K. Yoshigoe, I. Maljevic, I. Radusinovic, Software-defined fog network architecture for IoT, Wirel. Pers. Commun. 92 (1) (2017) 181–196.
[43] O. Consortium, et al., OpenFog Reference Architecture for Fog Computing, Architecture Working Group, 2017, pp. 1–162.
[44] N.K. Giang, V.C. Leung, R. Lea, On developing smart transportation applications in fog computing paradigm, in: Proceedings of the 6th ACM Symposium on Development and Analysis of Intelligent Vehicular Networks and Applications, 2016, pp. 91–98.
[45] P. Raj, A. Raman, Handbook of Research on Cloud and Fog Computing Infrastructures for Data Science, IGI Global, 2018.
[46] B.A.A. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, T. Turletti, A survey of software-defined networking: Past, present, and future of programmable networks, IEEE Commun. Surv. Tutor. 16 (3) (2014) 1617–1634.
[47] D. Kreutz, F.M.V. Ramos, P. Esteves Verissimo, C. Esteve Rothenberg, S. Azodolmolky, S. Uhlig, Software-defined networking: A comprehensive survey, Proc. IEEE 103 (1) (2015) 14–76.
[48] A.C. Baktir, A. Ozgovde, C. Ersoy, How can edge computing benefit from software-defined networking: A survey, use cases, and future directions, IEEE Commun. Surv. Tutor. 19 (4) (2017) 2359–2391.
[49] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, D. Sabella, On multi-access edge computing: A survey of the emerging 5G network edge cloud architecture and orchestration, IEEE Commun. Surv. Tutor. 19 (3) (2017) 1657–1681.
[50] P. Porambage, J. Okwuibe, M. Liyanage, M. Ylianttila, T. Taleb, Survey on multi-access edge computing for internet of things realization, IEEE Commun. Surv. Tutor. 20 (4) (2018) 2961–2991.
[51] D. Sabella, V. Sukhomlinov, Developing software for multi-access edge computing, ETSI WHite Paper No. 20, 2019.
[52] N. Makris, T. Korakis, V. Maglogiannis, D. Naudts, N. Nikaein, G. Lyberopoulos, E. Theodoropoulou, I. Seskar, C.A.G. Perez, M. Gomez, M. Tosic, N. Milosevic, S. Spirou, FLEX Testbed: a platform for 4G/5G wireless networking research, 2016.
[53] N. Fernando, S.W. Loke, W. Rahayu, Mobile cloud computing: A survey, Future Gener. Comput. Syst. 29 (1) (2013) 84–106.
[54] M.R. Rahimi, J. Ren, C.H. Liu, A.V. Vasilakos, N. Venkatasubramanian, Mobile cloud computing: A survey, state of art and future directions, Mob. Netw. Appl. 19 (2) (2014) 133–143.
[55] Y. Wang, Definition and categorization of dew computing, 3, (1) 2016, p. 7.
[56] 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.
[57] A. Ceselli, M. Premoli, S. Secci, Mobile edge cloud network design optimization, IEEE/ACM Trans. Netw. 25 (3) (2017) 1818–1831.
[58] D. Fernández-Cerero, J.Y. Fernández-Rodríguez, J.A. Álvarez-García, L.M. Soria-Morillo, A. Fernández-Montes, Single-board-computer clusters for cloudlet computing in internet of things, Sensors 19 (13) (2019) 3026.
[59] Q. Han, D. Cho, Characterizing the technological evolution of smartphones: insights from performance benchmarks, in: Proceedings of the 18th Annual International Conference on Electronic Commerce E-Commerce in Smart Connected World, ICEC '16, ACM Press, Suwon, Republic of Korea, 2016, pp. 1–8.
[60] V. Pejovic, M. Musolesi, Anticipatory mobile computing: A survey of the state of the art and research challenges, ACM Comput. Surv. 47 (3) (2015) 1–29, arXiv:1306.2356.
[61] J.-P. Hubaux, T. Gross, J.-Y. Le Boudec, M. Vetterli, Toward self-organized mobile ad hoc networks: the terminodes project, IEEE Commun. Mag. 39 (1) (2001) 118–124.
[62] U. Drolia, R. Martins, J. Tan, A. Chheda, M. Sanghavi, R. Gandhi, P. Narasimhan, The case for mobile edge-clouds, in: 2013 IEEE 10th International Conference on Ubiquitous Intelligence and Computing and 2013 IEEE 10th International Conference on Autonomic and Trusted Computing, IEEE, Italy, 2013, pp. 209–215.
[63] R.-A. Cherrueau, A. Lebre, D. Pertin, F. Wuhib, J.M. Soares, Edge computing resource management system: a critical building block! initiating the debate via OpenStack, in: USENIX Workshop on Hot Topics in Edge Computing, HotEdge 18, 2018, p. 6.
[64] D. Han, W. Chen, Y. Fang, A dynamic pricing strategy for vehicle assisted mobile edge computing systems, IEEE Wirel. Commun. Lett. 8 (2) (2019) 420–423.
[65] X. Huang, P. Li, R. Yu, Social welfare maximization in container-based task scheduling for parked vehicle edge computing, IEEE Commun. Lett. 23 (8) (2019) 1347–1351.
[66] P. Lea, in: J. Gonsalves (Ed.), IoT and Edge Computing for Architects, second ed., Packt Publishing Ltd., Birmingham, UK, 2020, p. 448.
[67] Y. Zhang, K. Guo, J. Ren, Y. Zhou, J. Wang, J. Chen, Transparent computing: A promising network computing paradigm, Comput. Sci. Eng. 19 (1) (2017) 7–20.
[68] 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) (2020) 1–36.
[69] I.I. Consortium, The Industrial Internet of Things Volume G1: Reference Architecture, Tech. Rep. 1.9, 2019.
[70] D.J. Rosenkrantz, S. Goel, S.S. Ravi, J. Gangolly, Structure-based resilience metrics for service-oriented networks, in: Dependable Computing, EDCC 5, in: Lecture Notes in Computer Science, vol. 3463, Springer Berlin Heidelberg, Berlin, Heidelberg, 2005, pp. 345–362.
[71] R. Yu, G. Xue, X. Zhang, Application provisioning in FOG computing-enabled internet-of-things: A network perspective, in: IEEE INFOCOM 2018 - IEEE Conference on Computer Communications, IEEE, Honolulu, HI, 2018, pp. 783–791.
[72] Z. Nezami, K. Zamanifar, K. Djemame, E. Pournaras, Decentralized edge-to-cloud load-balancing:Service placement for the internet of things, 2020, arXiv:2005.00270, arXiv:2005.00270 [cs].
[73] A. Brogi, S. Forti, A. Ibrahim, Deploying fog applications: How much does it cost, by the way?: in: Proceedings of the 8th International Conference on Cloud Computing and Services Science, SCITEPRESS - Science and Technology Publications, Funchal, Madeira, Portugal, 2018, pp. 68–77.
[74] 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, IEEE, Honolulu, HI, 2018, pp. 693–701.
[75] T. He, H. Khamfroush, S. Wang, T. La Porta, S. Stein, It's hard to share: Joint service placement and request scheduling in edge clouds with sharable and non-sharable resources, in: 2018 IEEE 38th International Conference on Distributed Computing Systems, ICDCS, IEEE, Vienna, 2018, pp. 365–375.
[76] L. Wang, L. Jiao, T. He, J. Li, M. Muhlhauser, Service entity placement for social virtual reality applications in edge computing, in: IEEE INFOCOM 2018 - IEEE Conference on Computer Communications, IEEE, Honolulu, HI, 2018, pp. 468–476.
[77] 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, IEEE, Madrid, Spain, 2017, pp. 89–96.
[78] B. Gao, Z. Zhou, F. Liu, F. Xu, Winning at the starting line: Joint network selection and service placement for mobile edge computing, in: IEEE INFOCOM 2019 - IEEE Conference on Computer Communications, IEEE, Paris, France, 2019, pp. 1459–1467.
[79] F. Faticanti, M. Savi, F.D. Pellegrini, P. Kochovski, V. Stankovski, D. Siracusa, Deployment of Application Microservices in Multi-Domain Federated Fog Environments, IEEE, Barcelona, Spain, 2020, p. 7.
[80] 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, IEEE, Las Vegas, NV, USA, 2019, pp. 1–6.

[81] 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, IEEE, Lisbon, Portugal, 2017, pp. 1222–1228.
[82] 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.
[83] T. Leppanen, C. Savaglio, L. Loven, T. Jarvenpaa, R. Ehsani, E. Peltonen, G. Fortino, J. Riekki, Edge-based microservices architecture for internet of things: Mobility analysis case study, in: 2019 IEEE Global Communications Conference, GLOBECOM, IEEE, Waikoloa, HI, USA, 2019, pp. 1–7.
[84] L. Zhao, J. Liu, Optimal placement of virtual machines for supporting multiple applications in mobile edge networks, IEEE Trans. Veh. Technol. (2018) 1.
[85] J. Furst, M.F. Argerich, B. Cheng, A. Papageorgiou, Elastic services for edge computing, in: Poster Session, IEEE, Rome, Italy, 2018, p. 5.
[86] M.F. Argerich, B. Cheng, J. Fürst, Reinforcement learning based orchestration for elastic services, 2019, arXiv:1904.12676, arXiv:1904.12676 [cs].
[87] M. Chen, W. Li, G. Fortino, Y. Hao, L. Hu, I. Humar, A dynamic service migration mechanism in edge cognitive computing, ACM Trans. Internet Technol. 19 (2) (2019) 1–15.
[88] A. Al-Shuwaili, O. Simeone, Energy-efficient resource allocation for mobile edge computing-based augmented reality applications, IEEE Wirel. Commun. Lett. 6 (3) (2017) 4.
[89] 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, IEEE, Valencia, 2017, pp. 74–80.
[90] E.F. Ordóñez-Morales, Vehicular fog computing on top of a virtualization layer, p. 15.
[91] P. Persson, O. Angelsmark, Calvin – merging cloud and IoT, Procedia Comput. Sci. 52 (2015) 210–217.
[92] E. Pournaras, P. Pilgerstorfer, T. Asikis, Decentralized collective learning for self-managed sharing economies, ACM Trans. Auton. Adapt. Syst. 13 (2) (2018) 1–33.
[93] M.S. de Brito, S. Hoque, T. Magedanz, R. Steinke, A. Willner, D. Nehls, O. Keils, F. Schreiner, A service orchestration architecture for Fog-enabled infrastructures, in: 2017 Second International Conference on Fog and Mobile Edge Computing, FMEC, IEEE, Valencia, Spain, 2017, pp. 127–132.
[94] W. Wong, A. Zavodovski, P. Zhou, J. Kangasharju, Container deployment strategy for edge networking, in: Proceedings of the 4th Workshop on Middleware for Edge Clouds & Cloudlets, MECC '19, ACM Press, Davis, California, 2019, pp. 1–6.
[95] J. Santos, T. Wauters, B. Volckaert, F. De Turck, Resource provisioning in fog computing: From theory to practice †, Sensors 19 (10) (2019) 2238.
[96] S. Hoque, M.S. De Brito, A. Willner, O. Keil, T. Magedanz, Towards container orchestration in fog computing infrastructures, in: 2017 IEEE 41st Annual Computer Software and Applications Conference, COMPSAC, IEEE, Turin, 2017, pp. 294–299.
[97] R. Perez de Prado, S. García-Galán, J.E. Muñoz-Expósito, A. Marchewka, N. Ruiz-Reyes, Smart containers schedulers for microservices provision in cloud-fog-IoT networks. challenges and opportunities, Sensors 20 (6) (2020) 1714.
[98] I. Stypsanelli, O. Brun, S. Medjiah, B.J. Prabhu, Capacity planning of fog computing infrastructures under probabilistic delay guarantees, in: 2019 IEEE International Conference on Fog Computing, ICFC, IEEE, Prague, Czech Republic, 2019, pp. 185–194.
[99] M. Noreikis, Y. Xiao, A. Yla-Jaaski, Qos-oriented capacity planning for edge computing, in: 2017 IEEE International Conference on Communications, ICC, IEEE, Paris, 2017, pp. 1–6.
[100] P. Pereira, J. Araujo, M. Torquato, J. Dantas, C. Melo, P. Maciel, Stochastic performance model for web server capacity planning in fog computing, J. Supercomput. (2020).
[101] 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, IEEE, Milan, Italy, 2019, pp. 19–27.
[102] H. Rafique, M.A. Shah, S.U. Islam, T. Maqsood, S. Khan, C. Maple, A novel bio-inspired hybrid algorithm (NBIHA) for efficient resource management in fog computing, IEEE Access 7 (2019) 115760–115773.
[103] O. Ascigil, T.K. Phan, A.G. Tasiopoulos, V. Sourlas, I. Psaras, G. Pavlou, On uncoordinated service placement in edge-clouds, in: 2017 IEEE International Conference on Cloud Computing Technology and Science, CloudCom, IEEE, Hong Kong, 2017, pp. 41–48.
[104] A. Tasiopoulos, O. Ascigil, I. Psaras, S. Toumpis, G. Pavlou, FogSpot: Spot pricing for application provisioning in edge/fog computing, IEEE Trans. Serv. Comput. (2019) 1.
[105] J. Tang, R. Yu, S. Liu, J.-L. Gaudiot, A container based edge offloading framework for autonomous driving, IEEE Access 8 (2020) 33713–33726.
[106] K. Poularakis, J. Llorca, A.M. Tulino, I. Taylor, L. Tassiulas, Joint service placement and request routing in multi-cell mobile edge computing networks, in: IEEE INFOCOM 2019 - IEEE Conference on Computer Communications, IEEE, Paris, France, 2019, pp. 10–18.
[107] S. Bi, L. Huang, Y.-J.A. Zhang, Joint optimization of service caching placement and computation offloading in mobile edge computing system, IEEE Trans. Wireless Commun. (2020).
[108] T. Subramanya, D. Harutyunyan, R. Riggio, Machine learning-driven service function chain placement and scaling in MEC-enabled 5G networks, Comput. Netw. 166 (2020) 106980.
[109] S. Bi, Y.J. Zhang, Computation rate maximization for wireless powered mobile-edge computing with binary computation offloading, IEEE Trans. Wireless Commun. 17 (6) (2018) 4177–4190.
[110] G. Lee, W. Saad, M. Bennis, An online framework for ephemeral edge computing in the internet of things, 2020, arXiv:2004.08640, arXiv:2004.08640 [cs, eess, math].
[111] Z. Zhang, L. Ma, K.K. Leung, L. Tassiulas, J. Tucker, Q-placement: Reinforcement-learning-based service placement in software-defined networks, in: 2018 IEEE 38th International Conference on Distributed Computing Systems, ICDCS, IEEE, Vienna, 2018, pp. 1527–1532.
[112] J.-y. Baek, G. Kaddoum, S. Garg, K. Kaur, V. Gravel, Managing fog networks using reinforcement learning based load balancing algorithm, 2019, arXiv:1901.10023, arXiv:1901.10023 [cs].
[113] M.G.R. Alam, M.M. Hassan, M.Z. Uddin, A. Almogren, G. Fortino, Autonomic computation offloading in mobile edge for IoT applications, Future Gener. Comput. Syst. 90 (2019) 149–157.
[114] Y. Qin, H. Wang, S. Yi, X. Li, L. Zhai, Virtual machine placement based on multi-objective reinforcement learning, Appl. Intell. (2020).
[115] L. Huang, S. Bi, Y.-J.A. Zhang, Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks, 2019, arXiv:1808.01977, arXiv:1808.01977 [cs].
[116] M. Tang, V.W.S. Wong, Deep reinforcement learning for task offloading in mobile edge computing systems, in: Proceedings of the 2019 4th International Conference on Mathematics and Artificial Intelligence, ICMAI 2019, ACM, hegndu, China, 2019, pp. 90–94.
[117] M. Bouet, V. Conan, Mobile edge computing resources optimization: A geo-clustering approach, IEEE Trans. Netw. Serv. Manag. 15 (2) (2018) 787–796.
[118] S. Song, C. Lee, H. Cho, G. Lim, J.-M. Chung, Clustered virtualized network functions resource allocation based on context-aware grouping in 5G edge networks, IEEE Trans. Mob. Comput. 19 (5) (2020) 1072–1083.
[119] K. Kaur, S. Garg, G.S. Aujla, N. Kumar, J.J.P.C. Rodrigues, M. Guizani, Edge computing in the industrial internet of things environment: Software-defined-networks-based edge-cloud interplay, IEEE Commun. Mag. 56 (2) (2018) 44–51.
[120] G. Zhang, F. Shen, Y. Yang, H. Qian, W. Yao, Fair task offloading among fog nodes in fog computing networks, in: 2018 IEEE International Conference on Communications, ICC, IEEE, Kansas City, MO, 2018, pp. 1–6.
[121] H. Shah-Mansouri, V.W.S. Wong, Hierarchical fog-cloud computing for IoT systems: A computation offloading game, IEEE Internet Things J. 5 (4) (2018) 3246–3257, arXiv:1710.06089.
[122] S. Jošilo, Task Placement and Resource Allocation in Edge Computing Systems (Doctoral thesis), KTH School of Electrical Engineering and Computer Science, Stockholm, Sweden, 2020.
[123] S. Jošilo, G. Dán, Selfish computation offloading for mobile cloud computing in dense wireless networks, 2016, arXiv:1604.05460, arXiv:1604.05460 [cs].
[124] A. Brogi, S. Forti, A. Ibrahim, How to best deploy your fog applications, probably, in: 2017 IEEE 1st International Conference on Fog and Edge Computing, ICFEC, IEEE, Madrid, Spain, 2017, pp. 105–114.
[125] K. Zhang, S. Leng, Y. He, S. Maharjan, Y. Zhang, Mobile edge computing and networking for green and low-latency internet of things, IEEE Commun. Mag. 56 (5) (2018) 39–45.
[126] S. Taherizadeh, A.C. Jones, I. Taylor, Z. Zhao, V. Stankovski, Monitoring self-adaptive applications within edge computing frameworks: A state-of-the-art review, J. Syst. Softw. 136 (2018) 19–38.
[127] R. Cziva, D.P. Pezaros, Container network functions: Bringing NFV to the network edge, IEEE Commun. Mag. 55 (6) (2017) 24–31.
[128] Google, cAdvisor repository.
[129] Prometheus Community, Node Exporter repository.
[130] Prometheus Community, Prometheus Monitor repository.
[131] J. Zhang, X. Zhang, W. Wang, Cache-enabled software defined heterogeneous networks for green and flexible 5G networks, IEEE Access (2016) 1.
[132] Q. Qin, K. Poularakis, G. Iosifidis, L. Tassiulas, SDN controller placement at the edge: Optimizing delay and overheads, in: IEEE INFOCOM 2018 - IEEE Conference on Computer Communications, IEEE, Honolulu, HI, 2018, pp. 684–692.
[133] N. Mohan, J. Kangasharju, Edge-Fog cloud: A distributed cloud for Internet of Things computations, in: 2016 Cloudification of the Internet of Things, CIoT, IEEE, Paris, France, 2016, pp. 1–6.

[134] B. Baek, J. Lee, Y. Peng, S. Park, Three dynamic pricing schemes for resource allocation of edge computing for IoT environment, IEEE Internet Things J. 7 (5) (2020) 4292–4303.
[135] H. Gupta, A.V. Dastjerdi, S.K. 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).
[136] 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, ACM Press, Austin, Texas, USA, 2017, pp. 47–52.
[137] S. Svorobej, P. Takako Endo, M. Bendechache, C. Filelis-Papadopoulos, K. Giannoutakis, G. Gravvanis, D. Tzovaras, J. Byrne, T. Lynn, Simulating fog and edge computing scenarios: An overview and research challenges, Future Internet 11 (3) (2019) 55.
[138] A. Markus, A. Kertesz, A survey and taxonomy of simulation environments modelling fog computing, Simul. Model. Pract. Theory 101 (2020) 102042.
[139] A. Brogi, S. Forti, QoS-aware deployment of IoT applications through the fog, IEEE Internet Things J. 4 (5) (2017) 1185–1192.
[140] 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, ACM, San Jose California, 2017, pp. 1–13.
[141] C. Adjih, E. Baccelli, E. Fleury, G. Harter, N. Mitton, T. Noel, R. Pissard-Gibollet, F. Saint-Marcel, G. Schreiner, J. Vandaele, T. Watteyne, FIT IoT-LAB: A large scale open experimental IoT testbed, in: 2015 IEEE 2nd World Forum on Internet of Things, WF-IoT, IEEE, Milan, Italy, 2015, pp. 459–464.
[142] 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: Cloud Computing and Services Science, Vol. 367, Springer International Publishing, Cham, 2013, pp. 3–20.
[143] N. Chen, Y. Yang, T. Zhang, M.-T. Zhou, X. Luo, J.K. Zao, Fog as a service technology, IEEE Commun. Mag. 56 (11) (2018) 95–101.
[144] 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, IEEE, 2019, pp. 1–6.
[145] G. Nguyen, S. Dlugolinsky, M. Bobák, V. Tran, A. López García, I. Heredia, P. Malík, L. Hluchý, Machine Learning and Deep Learning frameworks and libraries for large-scale data mining: a survey, Artif. Intell. Rev. 52 (1) (2019) 77–124.
[146] H.-J. Jeong, H.-J. Lee, S.-M. Moon, Cloud-based machine learning for IoT devices with better privacy: work-in-progress, in: Proceedings of the Thirteenth ACM International Conference on Embedded Software 2017 Companion, EMSOFT '17, ACM Press, Seoul, Republic of Korea, 2017, pp. 1–2.
[147] E.A. Castillo, A. Ahmadinia, Distributed deep convolutional neural network for smart camera image recognition, in: Proceedings of the 11th International Conference on Distributed Smart Cameras, ACM, Stanford CA USA, 2017, pp. 169–173.
[148] H. Li, K. Ota, M. Dong, Learning IoT in edge: Deep learning for the internet of things with edge computing, IEEE Netw. 32 (1) (2018) 96–101.
[149] J. Azar, A. Makhoul, M. Barhamgi, R. Couturier, An energy efficient IoT data compression approach for edge machine learning, Future Gener. Comput. Syst. 96 (2019) 168–175.
[150] L. Li, K. Ota, M. Dong, Deep learning for smart industry: Efficient manufacture inspection system with fog computing, IEEE Trans. Ind. Inf. 14 (10) (2018) 4665–4673.
[151] E. Li, L. Zeng, Z. Zhou, X. Chen, Edge AI: On-demand accelerating deep neural network inference via edge computing, IEEE Trans. Wireless Commun. 19 (1) (2020) 447–457.
[152] M. Lavassani, S. Forsström, U. Jennehag, T. Zhang, Combining fog computing with sensor mote machine learning for industrial IoT, Sensors 18 (5) (2018) 1532.
[153] F. Samie, S. Paul, L. Bauer, J. Henkel, Highly efficient and accurate seizure prediction on constrained IoT devices, in: 2018 Design, Automation & Test in Europe Conference & Exhibition, DATE, IEEE, Dresden, Germany, 2018, pp. 955–960.
[154] Y. Liu, C. Yang, L. Jiang, S. Xie, Y. Zhang, Intelligent edge computing for IoT-based energy management in smart cities, IEEE Netw. 33 (2) (2019) 111–117.
[155] F. Samie, L. Bauer, J. Henkel, From cloud down to things: An overview of machine learning in internet of things, IEEE Internet Things J. 6 (3) (2019) 4921–4934.
[156] F. Firouzi, B. Farahani, M. Barzegari, M. Daneshmand, AI-driven data monetization: The other face of data in IoT-based smart and connected health, IEEE Internet Things J. (2020).
[157] A. Nadian-Ghomsheh, B. Farahani, M. Kavian, A hierarchical privacy-preserving iot architecture for vision-based hand rehabilitation assessment, Multimedia Tools Appl. (2021) 1–24.
[158] I. Analytics, IoT security market report 2017–2022.
[159] M.S. Elbamby, M. Bennis, W. Saad, M. Latva-aho, C.S. Hong, Proactive edge computing in fog networks with latency and reliability guarantees, EURASIP J. Wireless Commun. Networking 2018 (1) (2018) 209.
[160] M.A. Lema, A. Laya, T. Mahmoodi, M. Cuevas, J. Sachs, J. Markendahl, M. Dohler, Business case and technology analysis for 5G low latency applications, IEEE Access (2017) 1.
[161] A. Nasrallah, A.S. Thyagaturu, Z. Alharbi, C. Wang, X. Shao, M. Reisslein, H. ElBakoury, Ultra-low latency (ULL) networks: The IEEE TSN and IETF DetNet standards and related 5G ULL research, IEEE Commun. Surv. Tutor. 21 (1) (2019) 88–145.
[162] H. Assasa, A. Loch, N.M. Prakash, R. Shyamsunder, S. Aggarwal, D. Steinmetzer, D. Koutsonikolas, J. Widmer, M. Hollick, Fast and infuriating: Performance and pitfalls of 60 GHz WLANs based on consumer-grade hardware, in: 2018 15th Annual IEEE International Conference on Sensing, Communication, and Networking, SECON, IEEE, Hong Kong, 2018, pp. 1–9.
[163] S. Aggarwal, A. Thirumurugan, D. Koutsonikolas, A first look at 802.11ad performance on a smartphone, in: Proceedings of the 3rd ACM Workshop on Millimeter-Wave Networks and Sensing Systems, MmNets'19, ACM Press, Los Cabos, Mexico, 2019, pp. 13–18.
[164] J. Sachs, G. Wikstrom, T. Dudda, R. Baldemair, K. Kittichokechai, 5G radio network design for ultra-reliable low-latency communication, IEEE Netw. 32 (2) (2018) 24–31.
[165] G. Pocovi, H. Shariatmadari, G. Berardinelli, K. Pedersen, J. Steiner, Z. Li, Achieving ultra-reliable low-latency communications: Challenges and envisioned system enhancements, IEEE Netw. 32 (2) (2018) 8–15.
[166] M. Liu, Y. Liu, Price-based distributed offloading for mobile-edge computing with computation capacity constraints, IEEE Wirel. Commun. Lett. 7 (3) (2018) 420–423.
[167] Z. Li, Z. Yang, S. Xie, W. Chen, K. Liu, Credit-based payments for fast computing resource trading in edge-assisted internet of things, IEEE Internet Things J. 6 (4) (2019) 6606–6617.
[168] IEEE Communications Society, IEEE Standard for Adoption of OpenFog Reference Architecture for Fog Computing, Tech. Rep., IEEE, ISBN: 9781504450171, 2018.
[169] M. Iorga, L. Feldman, R. Barton, M.J. Martin, N. Goren, C. Mahmoudi, Fog Computing Conceptual Model, Tech. Rep. NIST SP 500-325, National Institute of Standards and Technology, Gaithersburg, MD, 2018.
[170] 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, IEEE, 2017, pp. 89–96.
[171] 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, IEEE, 2017, pp. 1222–1228.
[172] A. Brogi, S. Forti, Qos-aware deployment of IoT applications through the fog, IEEE Internet Things J. 4 (5) (2017) 1185–1192.
[173] P.J. Sun, Privacy protection and data security in cloud computing: a survey, challenges, and solutions, IEEE Access 7 (2019) 147420–147452.
[174] C. Esposito, A. Castiglione, F. Pop, K.-K.R. Choo, Challenges of connecting edge and cloud computing: A security and forensic perspective, IEEE Cloud Comput. 4 (2) (2017) 13–17.
[175] J. Zhang, B. Chen, Y. Zhao, X. Cheng, F. Hu, Data security and privacy-preserving in edge computing paradigm: Survey and open issues, IEEE Access 6 (2018) 18209–18237.
[176] S. Parikh, D. Dave, R. Patel, N. Doshi, Security and privacy issues in cloud, fog and edge computing, Procedia Comput. Sci. 160 (2019) 734–739.
[177] I. Mohiuddin, A. Almogren, Security challenges and strategies for the IoT in cloud computing, in: 2020 11th International Conference on Information and Communication Systems, ICICS, IEEE, 2020, pp. 367–372.