



Orchestration in Fog Computing: A Comprehensive Survey

BRENO COSTA, JOAO BACHIEGA JR., LEONARDO REBOUÇAS DE CARVALHO, and ALETEIA P. F. ARAUJO, University of Brasilia, Brazil

Fog computing is a paradigm that brings computational resources and services to the network edge in the vicinity of user devices, lowering latency and connecting with cloud computing resources. Unlike cloud computing, fog resources are based on constrained and heterogeneous nodes whose connectivity can be unstable. In this complex scenario, there is a need to define and implement orchestration processes to ensure that applications and services can be provided, considering the settled agreements. Although some publications have dealt with orchestration in fog computing, there are still some diverse definitions and functional intersection with other areas, such as resource management and monitoring. This article presents a systematic review of the literature with focus on orchestration in fog computing. A generic architecture of fog orchestration is presented, created from the consolidation of the analyzed proposals, bringing to light the essential functionalities addressed in the literature. This work also highlights the main challenges and open research questions.

CCS Concepts: • General and reference → Surveys and overviews; • Computer systems organization → Cloud computing; Distributed architectures;

Additional Key Words and Phrases: Fog computing, orchestration, monitoring, resource management

ACM Reference format:

Breno Costa, Joao Bachiega Jr., Leonardo Rebouças de Carvalho, and Aleteia P. F. Araujo. 2022. Orchestration in Fog Computing: A Comprehensive Survey. *ACM Comput. Surv.* 55, 2, Article 29 (January 2022), 34 pages. <https://doi.org/10.1145/3486221>

1 INTRODUCTION

Cloud computing is already a mature paradigm that has been in place since 2006 [83]. It offers virtualized resources, created upon a huge shared infrastructure, that are consumed by the customers on a pay-per-use business model [139] and with a variety of cost options [19, 62]. Cloud computing is based on dozens of large datacenters, distributed around the world, that are placed in the center of the network and accessed by the Internet. Cloud computing also provides automatic scalability to its services.

Cloud computing datacenters are composed of thousands of resource-rich homogeneous physical servers that are interconnected by a redundant and stable network [157]. To optimize the infrastructure in use and comply with service and quality agreements made with the customers, a resource orchestration framework is in place.

Authors' address: B Costa, J. Bachiega Jr., L. R. de Carvalho, and A. P. F. Araujo, University of Brasilia, Department of Computer Science, Campus Darcy Ribeiro, Asa Norte, Brasilia, DF, 70910-900, Brazil; emails: {brenocosta, joao.bachiega.jr, leouesh}@gmail.com, aleteia@unb.br.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2022 Association for Computing Machinery.

0360-0300/2022/01-ART29 \$15.00

<https://doi.org/10.1145/3486221>

In recent years, with the technological advances in processors, memory and communications, a plethora of things was updated with computational capacity, shaping what is known as the Internet of Things (IoT) era [129]. Devices that measure health related parameters, such as smartbands, smart watches, smart building devices, smart city power grids, and traffic lights, are examples of this new area [50]. The increased computing power of mobile phones and the desired integration with these new smart devices create a scenario that combines together several heterogeneous devices working on the same application or service (e.g. a health monitoring app). Due to its aforementioned characteristics, cloud computing does not attend properly to these new computational demands, characterized by the need for a nearby computation paradigm.

Fog computing has emerged as a promising solution to meet this growing demand to expand the processing, network and storage capacity closer to end users, thus complementing the cloud computing fragility [11]. Fog computing's essential characteristics are low latency, large geographic distribution, heterogeneity, interoperability, real-time interactions and scalability [57]. However, researches on the use of this recent computational paradigm are still under development, with many challenges to overcome [131].

Among these challenges is fog orchestration, which aims to manage computational resources, dealing with heterogeneity, device mobility and connection uncertainty, to provide services to the customers while achieving **Service Level Agreement (SLA)** and **Quality of Service (QoS)** requirements [63].

In recent years, some research has been published on orchestration for fog computing, such as [32, 134, 150, 152]. However, all these papers have presented specific approaches to the orchestration and their proposals cover different areas, exposing the lack of consensus in the literature about what orchestration means in the context of fog computing. Viejo and Sánchez [144] stated that fog orchestration was still in its infancy since there were only a few published works that presented actual use cases.

This work aims to present a systematic and comprehensive review of the literature on orchestration for fog computing. To reach this goal, five research questions were defined to guide the analysis and to present relevant results to this topic. In addition, the challenges and open research issues will also be discussed. Thus, the main contributions of this work are:

- Define and delimit the scope of orchestration for fog computing;
- Present a generic architecture for fog orchestration, created from the consolidation of the 50 analyzed papers;
- Discuss some of the relevant challenges and open issues in the context of orchestration in fog computing.

To deliver these contributions, this work is organized as follows. Section 2 presents the main characteristics of fog computing and other related computing paradigms. Section 3 presents an assessment of orchestration history and its meaning in the context of this study. Section 4 compares this study with others surveys in the area. The methodology used for the systematic review and the research questions are presented in Section 5. A fog orchestration generic architecture, derived from the analysis of selected papers, is presented in Section 6. Next, the challenges inherent to the theme are presented in Section 7. Finally, Section 8 brings the conclusion and opportunities for future work.

2 FOG COMPUTING

Just as the natural phenomenon of fog is perceived when a cloud is close to the ground, the concept of fog computing was created to describe the computational paradigm that aims to bring the benefits of the cloud closer to end-devices. For this reason, it is considered a highly distributed

computing paradigm, highly integrated with the cloud, but with the processing also being carried out at the edge of the network, enabling the execution of applications that until then were not possible due to the high latency that existed in the communication between the devices and the cloud [91].

As fog computing concept is relatively recent, with its first definition presented in 2012 [11], there is still no consensus on the definition of fog computing, as well as requirements related to it, such as architecture, delivery methods, pricing model, etc. In this way, the initial definition of fog computing has been revised over time by several researchers from academia [22, 93, 138, 153] and industry [18, 56, 57, 101]. In these definitions it is possible to note that fog computing is closely linked to a cloud computing platform, since the fog computing's computational power will not completely replace the cloud when dealing with large volumes of data or complex processing problems [46]. Thus, fog computing is the right choice when the cloud cannot deliver to the applications and services the needed latency or runtime requirements.

2.1 Features and Architecture

According to **National Institute of Standards and Technology (NIST)** [57], the essential characteristics of fog computing can be summarized as:

- **Low Latency:** fog computing nodes are closer to the end users and can offer a faster analysis and response to the data generated and requested by the users, when compared to the operations made by a cloud service;
- **Geographic Distribution:** different from cloud computing, services and applications running on a fog computing infrastructure require geo-distributed deployment and management;
- **Heterogeneity:** allows the collection and processing of information obtained from different sources and collected by several means of network communication;
- **Interoperability and Federation:** resources must be able to interoperate with each other and services and applications must be federated across domains;
- **Real-Time Interactions:** fog computing services and applications involve real-time interaction, not just batch processing;
- **Scalability:** allows fast detection of variation in workload's response time and of changes in network and device conditions, supporting elasticity of resources.

Regarding fog computing architecture, NIST states [57] that one of its fundamental components is the fog node. Fog nodes are either physical components (e.g. gateways) or virtual components (e.g. virtual machines) that are tightly coupled with smart end-devices or access networks and provide computing resources to these devices. The fog nodes need to support one or more of the following attributes: autonomy, heterogeneity, hierarchical clustering, manageability, and programmability [57].

Fog computing environments can base their communication infrastructure on **Software-Defined Network (SDN)** [81], on **Radio Access Network (RAN)** [91] or on a composition of these technologies. Layered architecture is the most frequent representation used [1, 7, 22, 70, 93]. Specifically, a three-layered architecture is often used to represent a fog computing infrastructure [36, 71, 88]. However, some different proposals can be found in the literature:

- Four Layers [130]: (1) IoT, (2) Edge Nodes, (3) Intermediate Nodes and (4) Cloud;
- Five Layers [92]: (1) IoT, (2) Edge Network, (3) Intermediate Network, (4) Core Network and (5) Cloud;
- Six Layers [37]: (1) IoT, (2) Transport Layer, (3) Hardware Layer, (4) Algorithm Layer, (5) Regional Computing and (6) Cloud.

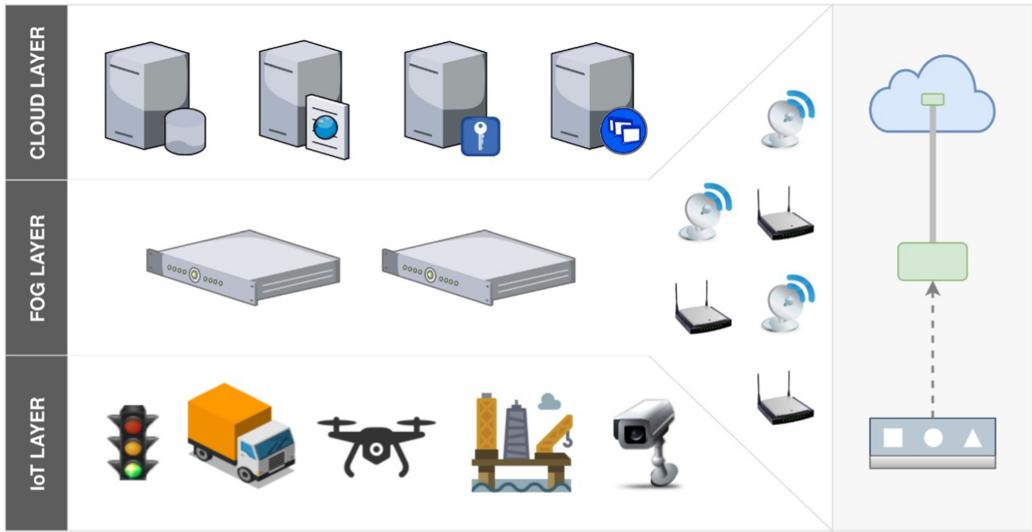


Fig. 1. Fog computing architecture overview.

Although there are variations in the number of layers in these proposed architectures, we can see that IoT and Cloud layers are present in all of them. Thus, the variations can be seen as different ways of structuring the Fog Layer. Thereby, regardless of the number of layers being proposed, the architectures can be summarized in three essential layers: IoT, Fog and Cloud, as presented in Figure 1. Unlike other proposals that used hierarchical architectures with hard borders, this work utilizes soft borders to delimit each layer, once fog computing devices can be found in the edge, near the IoT devices, and also near cloud computing devices, like telecom routers and switches. A comprehensive review of fog computing architectures can be found in [51].

The IoT Layer is a representation of all IoT devices connected at the the network edge. These devices can be operated by the end-users or collect environmental information autonomously. Data sent by them will be processed and stored in the Fog and Cloud layers. The Fog Layer is comprised of devices, called *Fog Nodes*. They are “smart” devices capable of processing, storing and routing data to the Cloud Layer [121], when needed. For some services that do not have high processing needs, the requests can be serviced directly from the Fog Layer, saving cloud resources and lowering the risk of network congestion on cloud communication channels. Additionally, it can function as a stage step between the IoT and Cloud layers providing the needed computational power for processing services and applications, such as data filtering and aggregation, before transferring the data to the cloud [4] when not enough resources are available on the Fog Layer. The Cloud Layer has more powerful resources to process the requests made by the devices in the IoT Layer, mainly those that were not fully answered by the Fog Layer. Thus, the Cloud Layer is a relevant component in a fog computing environment [4].

2.2 Related Computing Paradigms

There exist other related computational paradigms and technologies with analogous purposes to fog computing, but with diverse characteristics and architectures. For these paradigms, as with fog computing, there is still no mature definition in academia and this generates some confusion about the intersection among them.

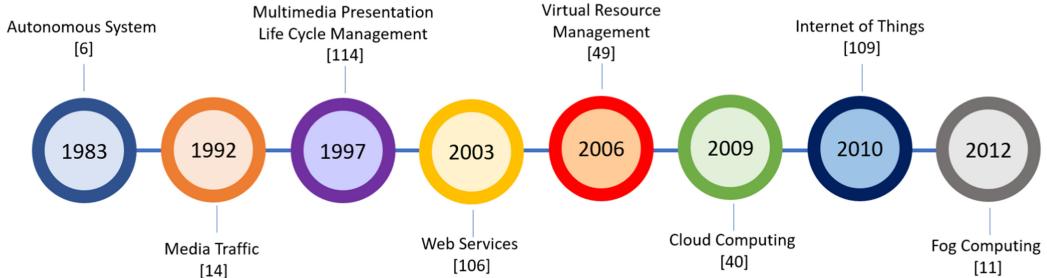


Fig. 2. Historical timeline of term Orchestration, adapted from [27].

In the literature review carried out for the development of this article, for example, it was possible to note that there was some confusion about fog computing, the edge computing and **Multi-access Edge Computing (MEC)** paradigms. Some authors consider fog and edge computing as synonymous [42, 123]. MEC can also be interpreted as a niche implementation of a fog environment [154] that is focused on mobility [140]. This happens due to the similarities between these computational paradigms, such as proximity to edge devices and also the need for integration with cloud computing to complement the computational resources required by applications, if necessary [157].

However, there are also differences between them. Fog computing runs services in a multi-layered architecture while edge computing runs services in a fixed logical location. Edge computing tends to be limited to a small number of peripheral devices [57], whereas fog computing, in general, has a larger number of devices. MEC can be defined as an implementation of edge computing to bring computational and storage capacities to the edge of the network within the **Radio Access Network (RAN)** to reduce latency and improve context-awareness [31].

In addition to Edge computing and MEC, it is also possible to find in the literature some variations and other proposals for paradigms and technologies that aim to perform tasks closer to the IoT devices: e.g., Mobile Cloud computing, Mist computing, Mobile Adhoc Cloud computing, Dew computing and Cloudlet Computing. In any case, supported by the definition of [127], “some other concepts, not declared as ‘fog computing’, might fall under the same ‘umbrella’”. In the literature there are several publications that have focused on the description and comparison of fog computing with these other computational paradigms and technologies, such as [91, 93, 157].

3 ORCHESTRATION

The Cambridge dictionary [13] defines orchestration as “a careful arrangement of something to achieve a particular result” and the Merriam-Webster [85] dictionary defines it as “harmonious organization”. The dictionaries bring the following synonyms to orchestration: managing, organizing, coordination, reorganization, and restructure.

In the work of de Sousa et al. [27], a survey about **Network Service Orchestration (NSO)**, the authors have presented a history of usage of “orchestration” in the literature related to NSO. Figure 2 shows a timeline of the first use of “orchestration” in different research areas. In Cloud computing, the first time this term was used was in 2009 and in fog computing the term appeared in 2012.

The paper [97] states that orchestration often refers to “a graph describing relationships between software elements or processes”, a definition also utilized by [39]. Accordingly, Viejo and Sánchez [144] define orchestration as the selection of nodes that will participate in the execution of a service or application. Once selected, these nodes communicate via a secure protocol that guarantees the

privacy of information exchange. It considers that the demanding entity, in this case an IoT device requesting services from the upper layer, will always be static and with a fixed association with the same fog/edge nodes, as in a factory structure. These cited papers have used orchestration as a dependency or composition graph. In this scenario, all entities involved have an obligatory participation. This utilization of “orchestration” is usual in the context of **Software Oriented Architecture (SOA)** and Web Services [106].

Other studies have definitions of orchestration that approach specific characteristics: orchestrator’s constituent modules, functionalities it is responsible for, and objectives it is supposed to reach. The paper [151] was one of the first articles that focused on orchestration in Fog environments. According to it, “Orchestration is a key concept within distributed systems, enabling the alignment of deployed applications with users’ business interests”. In its vision, the orchestrator must “predict, detect, and resolve issues pertaining to scalability bottlenecks that could arise from increased application scale”. In the words of [60], orchestration “refers to the processes of managing and coordinating the physical computational resources provided by the underlying infrastructure to serve the applications”. According to [24], “nowadays, orchestration is an overused word; it is presented in different scenarios to indicate the management of the life cycle of a one or more distributed components that together deliver a service or functionality”.

The current work uses “orchestration” as a synonym of management, arrangement and coordination. Using the dictionary definitions presented before, it is important to be careful and consider specific characteristics of fog components that will participate in the orchestration. Besides, there is a need to be harmonious, balancing eventual opposing objectives (e.g. lower latency vs cost reduction, energy saving vs reaching SLAs) in order to achieve planned results. Two other synonyms, restructure and reorganization, give dynamism to orchestration in a way that could envision a lively adaptive or autonomous process.

Considering the diversity of concepts presented, it is necessary to assign a definition for the term “orchestration” in the fog computing context, and because there is still no standardization adopted by the academy, the following definition is proposed: “Fog orchestration is a management function responsible for service life cycle. To provide requested services to the user and assure the SLAs, it must monitor the underlying infrastructure, react timely to its changes and comply with privacy and security rules”.

3.1 Orchestration on Fog Computing Related Areas

Orchestration is also found in fog computing related areas that are outside of this work’s scope, as container orchestration, **Software Defined Network (SDN)** and **Network Function Virtualization (NFV)**. Figure 3 shows the percentage of results found for each area when searching for fog orchestration in the databases described in the Section 5.

3.1.1 Software Defined Network (SDN) and Network Function Virtualization (NFV). Software Defined Network (SDN) [126] and Network Function Virtualization (NFV) [53] are network technologies in evidence on the scenario of **future internet (FI)** due to their improvements to network programmability and virtualization [133]. SDN’s main characteristic is to separate control and data planes that are tightly coupled on classic networking devices. Due to this, SDN can distribute network control functions without losing a centralized (logically) network view in a single point of management. NFV’s target is to simplify the use of **virtual network function (VNF)**, a software-based network service, on general-purpose hardware. NFV’s benefits are cost reduction and elasticity improvements to network functions [133].

They are technologies designed to evolve the network layer, expanding network availability and performance through the **management and orchestration (MANO)** of network services.

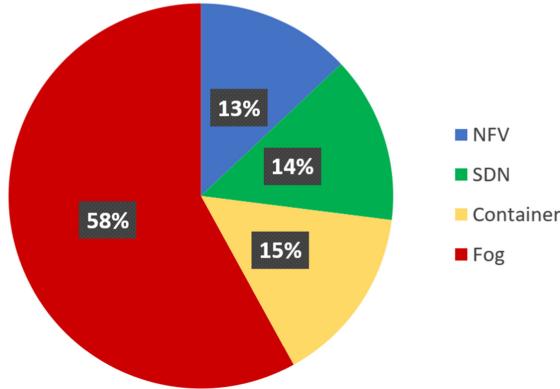


Fig. 3. Percentage of results from out of scope areas when searching for fog orchestration.

But SDN controllers lack common application interfaces (northbound interfaces) and NFV orchestrators rely on different infrastructure models [76]. According to Toosi et al. [133], both Fog and SDN/NFV are in their infancy and there is no significant research on their integration yet.

This survey aims to delve into fog orchestration, a process that is executed on the application layer. Despite these differences, Lingen et al. [136] proposed an architecture that integrates the orchestration of network services with SDN/NFV and user services (applications) on the same framework. It is one of the 50 analyzed papers, described in the Section 6.5.

3.1.2 Container Orchestration. In an environment such as fog computing, composed potentially of resource-restricted devices [157], the use of containers, a lightweight and small footprint form of virtualization [55], is referred to as being more appropriate than the use of virtual machines as execution environments. There are several container runtimes and Docker [29] is the most well-known one. Besides the runtime itself, a container orchestration solution, i.e. Kubernetes [67], Docker Swarm [30], Nomad [100] and Marathon on Mesos [77], can be used to manage container life cycle, scale them up or down, do self-healing, migrate them and manage a heterogeneous infrastructure, abstracting the physical devices where they execute. Thus, container orchestration solutions can be adopted to implement and automate some of the essential characteristics of Fog Computing (Section 2): heterogeneity, geographic distribution, and scalability.

On the other hand, they have limitations that make them ill-adapted to be fog orchestration solutions, such as those studied in this work [32]. In a service's orchestration scenario, the functionalities delivered by deployed artifacts and the relationship among them are central, but in a container's orchestration scenario this is not always true [24]. Containers are not device-aware and they do not consider the geo-location of service instances when deciding which one should handle each request [44]. Thus the deployment of a container cannot be enforced to a node that has a specific characteristic (device or sensor attached) and there is no guarantee that the users will be serviced by a nearby replica [24], [35]. Conversely, a container management solution could favor resource-rich devices, even if they are located in the cloud, due to their better performance and load metrics, and this behavior can increase communication latency, harming an essential characteristic of fog computing, as described in Section 2. Finally, container orchestration solutions implement only a part of what this paper has proposed as fog orchestration on previous subsection. They do not deliver a full service life cycle management and compliance with privacy and security rules [135]. Despite these limitations, a container orchestration solution can be a component of a broader fog orchestration framework [152], [69] as described in the Section 6.5.

4 RELATED WORK

Some papers presented taxonomies and surveys about orchestration on different areas of study, for example SDN [66] and NFV [86]. In [149], the authors proposed a taxonomy for cloud resources and conducted a survey of cloud orchestration techniques. In the area of **Network Service Orchestration (NSO)**, an overview of on-going efforts from research projects, standardization bodies, and both open-source and commercial implementations of MANO frameworks is presented in [27]. These papers have not included fog computing as a subject of study.

Vaquero et al. [137] approached challenges related to orchestration within which they have named new generation technologies like fog computing, SDN, NFV, **Function as a Service (FaaS)**. The authors analyzed the functional orchestration needs brought by these new paradigms and the requirements that must be met by a next generation of orchestrators that want to implement them. Several findings and proposals made were designed for paradigms other than fog computing (e.g. SDN, FaaS) and their applicability to fog is questionable or even impossible. This work has only analyzed orchestration proposals that can be implemented on fog computing.

The paper [107] surveyed the literature to find and present the main functionalities of fog computing on the context of smart cities applications. The authors mainly focused on connectivity and device configuration aspects of fog computing, approaching only a few orchestration functionalities: resource management, communication management and security.

The paper [113] compared fog computing to other paradigms (Transparent Computing [153], MEC [31] and Cloudlet [122]), presenting similarities and differences. The authors also have discussed about offloading, caching, security and privacy on these paradigms, but without approaching orchestration functionalities.

The survey [72] approached application management on fog computing and identified three main aspects of it: application architecture, application placement and application maintenance. The authors presented a taxonomy for each of them. Although the taxonomies have covered some fog orchestration functionalities, other important ones were outside the scope, e.g. request admission, communication management and fog infrastructure related monitoring and resource management.

The paper [98] built a systematic mapping study of **Deployment and Orchestration Approaches for IoT (DEPO4IOT)**. The authors use orchestration as a synonym of service composition instead of a broader coordination and management function that includes service composition and aggregates several other functionalities as this survey does.

The authors of [112] surveyed the literature and presented fog computing main characteristics, common application domains, existing software and hardware platforms for the IoT and research challenges. Fog computing orchestration was detailed and described as one of these challenges, but its functionalities were discussed only partially.

In the work of [140], the authors presented research challenges of fog orchestration and did a survey about cloud, fog, edge and MEC orchestration architectures. Although 75 works have been selected for analysis, only four of them had their proposed architectures compared.

In contrast, this work presents an analysis of 50 fog orchestration proposals and it has not included cloud orchestration due to the relevant differences that a centralized paradigm has when compared to distributed ones. Besides, this work aims to carry out a comprehensive and updated orchestration review and bring to light the main ideas and solutions proposed in the literature, being an unbiased source of information about fog computing orchestration. Table 1 shows a comparison between the related work presented in this section and the contributions made by this work. As noted, the aforementioned surveys did not delve deeply into fog computing orchestration concepts, architecture and functionalities.

Table 1. Related Work Comparison

Surveys	Year	Items				
		Focus on Fog Computing	Conceptualizes orchestration	Discusses orchestration functionalities	Consolidates orchestration architectures	Relates orchestration to management
Kreutz et al. [66]	2014			∅		
Mijumbi et al. [86]	2015		∅			✓
Weerasiri et al. [149]	2017		∅	✓	✓	✓
Perera et al. [107]	2017	✓		∅		
Velasquez et al. [140]	2018	✓	✓	✓		✓
De Sousa et al. [27]	2019		✓	✓		✓
Vaquero et al. [137]	2019	∅		✓		✓
Ren et al. [113]	2019	∅		∅		✓
Nguyen et al. [98]	2019	✓	∅	∅		
Puliafito et al. [112]	2019	✓	✓	∅		✓
Mahmud et al. [72]	2020	✓		✓		
This work	2021	✓	✓	✓	✓	✓

✓ denotes comprehensive discussion about the item.

∅ denotes partial or superficial discussion about the item.



Fig. 4. Steps of SLR methodology used [65] and [108].

5 RESEARCH METHODOLOGY

The methodology of **Systematic Literature Review (SLR)** adopted in this work was based on the works [65] and [108]. The steps for the development of this research are summarized in Figure 4 and they will be detailed in the next subsections.

5.1 Research Questions

In the first stage of an SLR, **Research Questions (RQ)** must be defined, helping to determine what is being researched, which results should be achieved and guide the SLR [65].

For the purpose of understanding how fog orchestration is characterized in the literature, five RQs were formulated:

- **RQ1: What are the goals of orchestration?** - this question aims to collect the proposal's objectives, exposing which fog computing issues were intended to be solved. The answers to this question will be compared to the answers of RQ5 to identify the trends in fog orchestration.
- **RQ2: What are the orchestrated entities?** - the answer to this question will show the variety of terms used in the literature and help to disambiguate their meaning. Besides, they

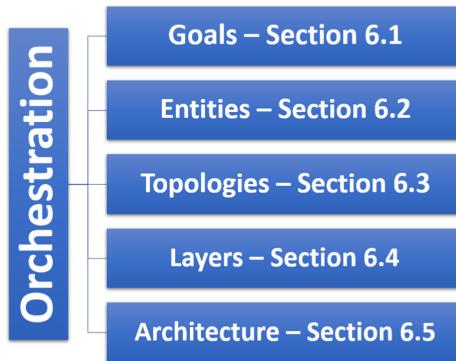


Fig. 5. Research questions overview.

will support the concept of orchestration as a complex management function and not a sub area of resource management.

- **RQ3: What is orchestrator control topology?** - this research question will help to categorize the proposals according to the control flow adopted to make and disseminate decisions. Analyzing the answers, it can be easier to understand the issues that can arise from the approaches used and to propose solutions already validated on other distributed paradigms.
- **RQ4: Which architecture layers are considered?** - considering fog computing architecture presented in Figure 1, the answers to this question will show the trends in fog computing and the relevance of communication and fault management to accomplish orchestration's goals.
- **RQ5: What are the functionalities comprehended by the orchestration?** - orchestration is a management and coordination function as conceptualized in the Section 3. This question helps to uncover orchestration proposals and to consolidate a generic architecture from the ones found in the state of the art.

These questions were also the starting point for search string elaboration, as well as to assist in the assessment of publications. After performing the search and reading the selected papers, we expect to find the answers to the RQs. Figure 5 presents an overview of research questions and points out the subsections where each one will be discussed.

5.2 Search Process

The search process included the selection of the online search databases that were used, the period and the search string elaboration. For this article, Scopus,¹ Web of Science,² ACM Digital Library³ and IEEE XploreLibrary⁴ databases were used as research sources. The following search string was used:

“orchestrat AND (fog OR edge OR mec)”*

This string was built aiming to bring results that could answer the research questions, presented in Section 5.1. Besides “orchestration” we also target the variations “orchestrating”, “orchestrated” and “orchestrator”. As explained in Section 2, some authors consider fog and edge computing as

¹scopus.com.

²webofknowledge.com.

³dl.acm.org.

⁴ieeexplore.ieee.org.

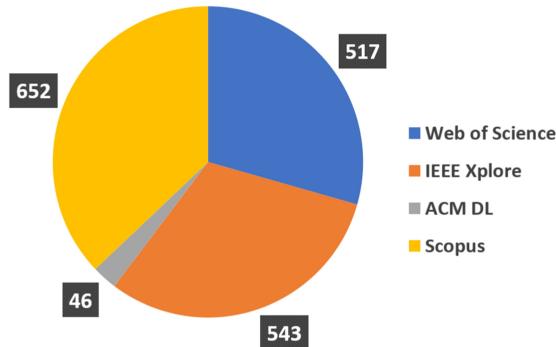


Fig. 6. Returned results by database.

synonymous and others define MEC as being a niche interpretation of fog computing, focused on mobility. Due to these considerations, edge and MEC were included in the search string.

5.3 Inclusion and Exclusion Criteria

After using the search string in the related online databases, it was necessary to apply the inclusion and exclusion criteria. Therefore, the inclusion criteria defined for this research were:

- Publications written in English;
- Peer-reviewed publications;
- Publication date starting from 2012 (year of the first publication on fog computing [11]);
- Publications that present architectural models, techniques or methods applied to orchestration in fog computing.

Likewise, exclusion criteria were those that do not meet any of the inclusion criteria listed above, as well as:

- Duplicated publications among the databases;
- Publications that are focused on the network layer;
- Publications that present already analyzed orchestration proposals. In this case, the most recent publication will be included in the survey;
- Publications that have focused on only one of the orchestration functionalities (e.g. resource management or monitoring) instead of on the coordination and integration of them.

5.4 Quality Assessment

Initially, the research carried out in the four selected databases using the search string and the inclusion and exclusion criteria returned a total of 1,758 publications. The quantity of search results by database can be viewed in Figure 6. After removing the duplicates, the number of 1,758 was reduced to a total of 1,066 publications. Screening of titles and abstracts reduced the quantity to 130 publications.

After reading a sample of selected works, we could identify the most frequently described orchestration's functionalities, i.e. resource management, monitoring and security. To prevent excluding relevant works that have not used "orchestration" as a management function that coordinates those functionalities, we decided to run the query defined in Section 5.2 again, but substituting "orchestrat*" for the functionalities' names. This step added 15 other publications, reaching a total of 145. The last step was to read the full text of those 145 works, selecting the 50 publications that will be analyzed in this survey.

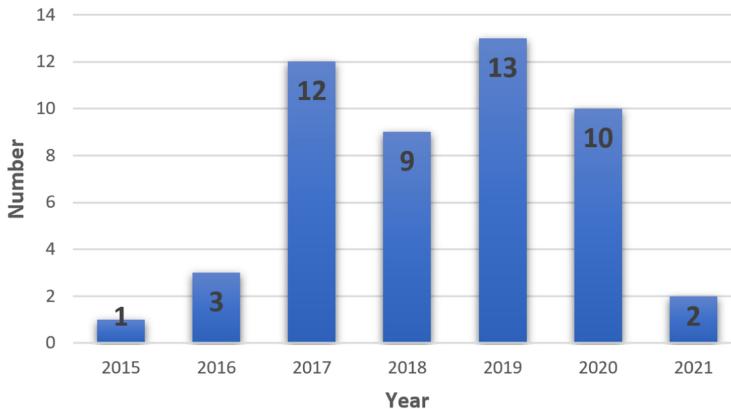


Fig. 7. Publications by year.

5.5 Data Collection

The first analysis of the 50 selected publications was related to their distribution over the years. Figure 7 shows this information. It indicates that, although the subject has received more attention only from 2017, the interest in it remained constant through the years until now. It is important to notice that 2021 was considered only until February.

A full text reading and analysis of each one of the 50 selected papers was performed to help to answer the five research questions. The results are presented in the next section.

6 RESULTS

This section presents the results obtained from the 50 selected publications, related to orchestration on fog computing.

6.1 Goals of Orchestration

RQ1 aims to determine what the goals are that the analyzed orchestration's proposals try to reach. Orchestration itself is a coordination of functions and processes and its goals can be perceived as a resultant of the goals of its constituents. Nevertheless, several papers presented explicit goals they tried to reach with their proposals and they are described in this section. Some papers have presented more than one goal and, in this case, they may appear in more than one category. The analyzed papers that were not cited in the next paragraphs have not presented any specific goals for their orchestration proposal. Table 2 summarizes the most frequent goals presented by the analyzed papers and defines a mnemonic code for each goal for further reference.

A proper and optimized resource management is one of the most cited goals [9, 16, 21, 32, 44, 52, 61, 80, 118, 119, 125, 148, 155, 158]. Indeed, processes like service placement and computation offloading have a strong dependency with resource availability [73]. Another frequently cited goal is the guarantee of **Service Level Agreements (SLAs)**. Some papers refer to this goal as latency reduction [59, 142]. Others as an improvement on application performance [58] or application response time [95], and several set the goal as guaranteeing SLAs or **Quality of Service (QoS)** [44, 102, 119, 141, 145, 147, 155]. Service life cycle management is another goal cited by some papers [20, 28, 47]. It refers to the process of enrolling a service, storing its images on a repository, delivering this image to an executing platform when requested, monitoring the availability and taking proper actions when necessary (e.g. update the image, migrate or replicate the service), attending to requirements previously agreed. Last but not least, there are, among the papers

Table 2. Most Frequent Goals Cited by Analyzed Papers

Goal	Code	Analyzed Papers	Total
Resource management	RM	[9, 16, 21, 32, 44, 52, 61, 80, 118, 119, 125, 148, 155, 158]	14
Guaranty of SLAs	SLA	[44, 58, 59, 95, 102, 119, 141, 142, 145, 147, 155]	11
Service life cycle management	SLM	[20, 28, 47]	3
Minimizing energy consumption	ENE	[58, 147]	2
Improving application resilience	APP	[111, 141]	2
Lowering costs	COS	[34, 147]	2
Providing a trusted orchestration	SEC	[103]	1
Converging NFV, 5G and Fog	NFV-5G	[136]	1

analyzed, also citations to other specific goals as lower costs [34, 147], minimizing energy consumption [58, 147], improving application resilience [111, 141], providing a trusted orchestration [103], and implementing the convergence of NFV, 5G and Fog [136].

Knowing the goals of a specific proposal can help the researcher to better understand the technical choices and architectural decisions made by the authors of that proposal. Analyzing the results summarized in Table 2 it is possible to understand why fog orchestration proposals have RM as the main functionality (48 of 50 analyzed proposals have implemented it, as can be seen in Section 6.5). RM as goal have appeared more frequently in years 2017 and 2018 [16, 118, 119, 148, 155, 158] and SLA/SLM, in 2019 and 2020 [20, 28, 47, 58, 59, 95]. This finding can suggest a changing in fog orchestration perspective, in the period of time comprehended by this analysis, from mainly managing fog infrastructure to run the services to a management and coordination function, managing mainly user relationship with the services provided. As the period of analysis is short, it is important to monitor the literature to see if this is an actual trend.

6.2 Orchestrated Entities

The second **Research Question (RQ2)** to be answered refers to the main components being orchestrated by the frameworks and architectures proposed. The components are referred here as orchestrated entities and their frequency of appearance among the analyzed papers can be seen in Figure 8.

The most frequent entity cited as the target of orchestration actions, among the analyzed publications, is service [15, 23, 24, 103, 104, 118, 134, 136, 141, 159]. It is a generalization of a piece of software that delivers a result and must be executed on the fog platform. Some papers have cited this entity as service components, service segments or micro-services [12, 60, 80, 95, 111], bringing the idea of a fine-grained piece of software. Others referenced them as service chain or service composition [32, 142, 147], showing the need for managing the dependencies among them.

Within the context of fog orchestration, a synonym of service is application. This orchestrated entity is referred by several papers [5, 34, 52, 61, 99, 102, 120, 148, 150, 156, 158]. A handful of papers use application components with the same meaning of micro-services, also exposing the necessity of managing the dependencies and the composition of small parts to build a larger and meaningful piece of software [44, 47, 69, 87, 119, 152].

Another orchestrated entity found was task [9, 16, 21, 58, 59, 82, 110, 125, 132]. It has also appeared as a composition of tasks, a pipeline [3] or a workflow [145].

Other entities were found related to specific proposals. The authors of [20] have used pipelines of “edge functions” (portions of code that will execute on fog nodes). In [155], the entity is a “moving query”, a request of service made by an IoT object and targeted to another moving IoT object. On [28] they have used “analytical pipelines”. Only one paper used “hardware”, mobile edge devices, as a possible orchestrated entity [132]. The authors of [118] also cited the orchestration of end

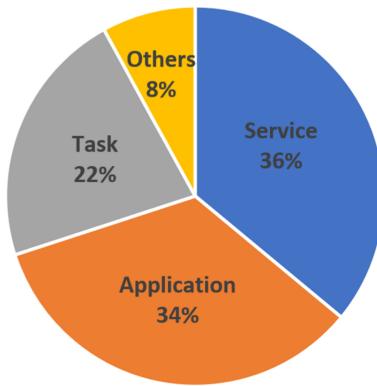


Fig. 8. Orchestrated entities.

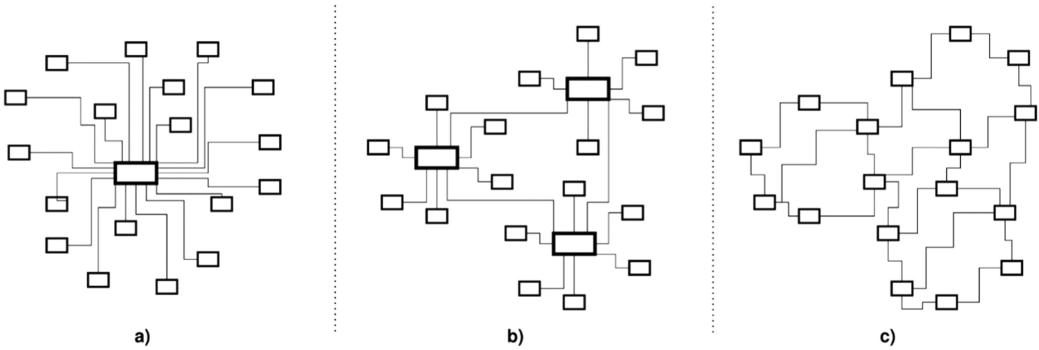


Fig. 9. Orchestrator's control topology. (a) Centralized; (b) Decentralized; (c) Distributed. Adapted from [79].

devices, but the actual orchestration on their proposal is only about resources and services. As each of these entities was referred by only one paper, they were grouped under the label “Others” in Figure 8.

It is not the intention of this paper to discuss the minor differences in the meaning of the orchestrated entities used by analyzed papers. It is helpful to present all the different names used in the literature to inform the fog computing orchestration researcher that these differences are not meaningful from the point of view of an orchestration framework, a management and coordination function. In this sense, the word “service” will be used from now on, representing an executable piece of code that is managed by the orchestration framework in place.

6.3 Orchestrator Control Topology

RQ3 intends to identify which control topologies are used to implement the orchestrator concerning the control flow and decision making. Masip et al. [79] described three control topologies that can be used in fog environments: centralized, with only one central node as the orchestrator; decentralized, where groups of nodes have a local leader node and the leaders collaborate on decision making in a pre-defined way; distributed, where all the nodes can interact to make decisions about the whole infrastructure. A visual representation of these topologies can be seen in Figure 9.

6.3.1 Centralized. A large number of proposals, 34 of 50 papers, have proposed a centralized topology. With an orchestrator at a central position it is possible to achieve a comprehensive view

of the distributed and dynamic infrastructure. Thus, functions as resource management become simpler in this scenario [5, 12, 16, 20, 23, 28, 32, 34, 47, 52, 69, 80, 82, 84, 87, 95, 99, 102–104, 111, 119, 120, 125, 134, 136, 141, 145, 147, 150, 152, 156, 158, 159].

On the other hand, the problem of having a **Single Point of Failure (SPOF)** and its consequences should be addressed. Some papers have proposed solutions to this issue (SPOF): the use of replication jointly with leader election. In this sense, although there is only one orchestrator at a time, there is a predefined way to choose another in case of orchestrator's failure or unavailability [103, 111, 145].

In [148], each service has a centralized server in the cloud that supplies the requests made by the users. To improve service performance, a partitioned server can be placed on fog nodes so the users in the vicinity can connect to this local server and have their service-related data managed locally instead of connecting to the cloud. Periodically, the fog nodes update the centralized server so it can have a global view of the service usage. When a fog node cannot execute an additional local server or cannot guarantee the SLAs, it denies the allocation or, when already allocated, it migrates local data back to the cloud.

6.3.2 Decentralized. A handful of papers have proposed a decentralized topology for decision making regarding the orchestrator's functions. Some functionalities can be distributed among several nodes. These nodes have to exchange status information about the resources they manage and they have to make decisions using the available information, that can be partial.

In the proposal published in [60], the virtualized infrastructure is splitted into domains, i.e. groups of nodes, and one of these nodes is the controller, locally responsible for the orchestration of their resources. Controllers communicate to each other using **peer-to-peer (P2P)** protocol. In the architecture proposed by [24], there is a central entity, called **Fog Orchestrator (FO)**, that delivers actions to be executed by distributed entities, called **Fog Orchestrator Agents (FOA)**. FOAs can act also as a central entity when needed: e.g. in the case that the FO can not be reached due to connectivity shortages. In the approach used by [142], a centralized strategy takes action on northbound communication and a decentralized one on the southbound communication. Choreography is used on the southbound communication. In [155], an entity called midway server, located on the Fog layer, is in charge of a geographical region of interest. When a request arrives, e.g. a query about the load of traffic in the next five miles, it broadcasts the query to the edge gateways, located on the IoT layer and in charge of tracking the things (vehicles, road side units, mobile phones) in its local area. Each edge gateway has an installed query table used to notify the midway server when a device goes outside the range of each query. This decentralized architecture aims to reduce communication cost considering that the things are moving objects.

In Lamen [118], the central orchestration module is in the cloud and it is responsible for receiving the service requests and delivering them to the decentralized modules in the Fog, called mediators. Mediators are responsible for the discovery and resource management of a cluster of IoT devices, gathered together based on their geo-location. When mediators receive a request for service execution it selects the devices that will do the job and delivers the result back to the cloud, keeping private the devices' identities. In the cloud, results are summarized and returned to the requester.

In [44], there are centralized application controllers in the cloud that manage application placement and life cycle. A specific process runs on the cloud to automatically discover new fog sites, groups of fog nodes that can share resources from the devices that compose them. At each fog site, there is an agent responsible for site tenancy, resource management and monitoring. According to application policies in the cloud, an application controller can share resources from more than one fog site and this flexibility improves application performance.

In Hydra [61], each node can assume one or more roles per application deployed on the system, e.g. leader, host. Using a location-aware based algorithm, a specific node is determined as the leader of that application and a group of nodes is set to host and control that application in the system. In a per application basis, there are several groups of nodes, each one with a leader, that interact to acquire and release resources, i.e. nodes, as needed.

Only [3] have published no details about the way it works and which functions were addressed on the decentralized architecture the authors have proposed.

6.3.3 Distributed. In the last orchestrator control topology, there is no central or leader node. All nodes can make decisions about orchestration functionalities and communicate the results to the others. In [58], each node decides whether and where to offload tasks (to itself, to other nodes or to the cloud), according to the availability of resources. In [132], each fog device does task orchestration. On a rotation-basis, one of the devices does area orchestration, moving the mobile devices in a way that more computational resources be placed closer to where more requests are coming. In the case of [15], there is a **Service Defined Orchestrator (SDO)** for each deployed service. SDOs use a consensus algorithm to acquire and allocate needed resources from the shared infrastructure. In the FORA platform [110], an application deployment starts with the collaboration between fog nodes. Each node can make local decisions about which application tasks it will execute. To better decide, a node receive information about resources and communication latency of other neighbor nodes.

Although Centralized is the most used control topology among analyzed papers, only three of them have addressed the SPOF risk. This shows that fault tolerance needs to be improved in fog orchestration. Besides, as heterogeneity is one of the essential characteristics of fog computing, as seen in Section 2, it is expected that the node running the orchestrator be one of the most powerful nodes available, that high availability strategies be used in a set of nodes (clustering, replication, etc.) or even that the orchestrator executes on cloud nodes, but these scenarios were barely exploited by analyzed papers. Fog Computing is a distributed paradigm. One may think that the proper control flow topology would also be a distributed approach, but this assumption is not confirmed by the findings in this survey. To implement decentralized or distributed decision control flow, there is a complexity increase due to the need for consensus management and data replication, processes that are more dependent of good communication channels and higher processing power, although connectivity issues and restricted resource nodes are expected on fog infrastructures.

6.4 Covered Architecture Layers

The analysis of which architecture layers are considered in orchestration approaches in fog computing is necessary to answer RQ4. Fog extends the cloud, providing computational resources to execute services on the edge, closer to the end users. As described in Section 2, this paper considers a three layered architecture (1.Cloud, 2.Fog, and 3.IoT) where the Fog layer includes everything between the cloud and the end users. This section categorizes the analyzed papers according to the layers where their proposals orchestrate services on.

6.4.1 Fog Layer. A handful of papers have proposed frameworks to orchestrate services considering only the Fog layer. A representation of it can be seen in Figure 10(a). The resource management function (see Section 6.5.3) will have to deal with heterogeneous devices, but it is expected that they occur in a moderate quantity when compared to the IoT layer. This scenario has a lower complexity when compared to multi-layer scenarios as it will be described on the next subsections. Even when there is a need for cloud resources, i.e. lack of proper fog resources required by a

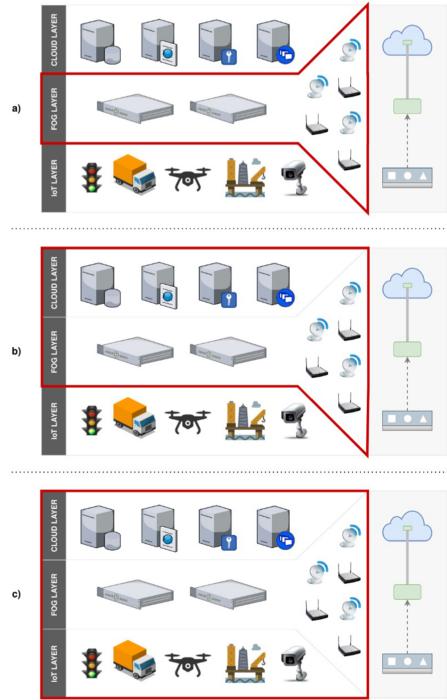


Fig. 10. Fog architecture layers where orchestration operates (inside red lines): (a) only Fog Layer; (b) Fog and Cloud Layers; (c) all Layers.

service, they are requested by fog orchestration by means of a cloud provider's APIs. The following papers have proposed orchestration in the Fog layer: [3, 20, 23, 24, 32, 69, 99, 111, 147, 158].

6.4.2 Fog and Cloud Layers. Some analyzed papers have considered not only the Fog layer, but also the Cloud layer as places where services could be executed. A representation of this scenario can be seen in Figure 10(b). In these cases, fog orchestration must monitor and manage available resources on two different layers and there is a complexity increase in this management. The papers are: [5, 12, 16, 21, 34, 52, 59, 84, 87, 95, 110, 125, 132, 134, 141, 148, 159].

6.4.3 IoT, Fog and Cloud Layers. Orchestrating services on all layers differs from the last presented scenario due to the use of IoT devices' resources. Having the IoT layer as a possible execution environment means that the end user's devices must be managed and share their available resources. A representation of this scenario can be seen in Figure 10(c). A huge number of devices are expected in this layer [43], and this will increase management complexity. The following papers have proposed orchestration functions that manage services throughout all layers: [9, 15, 28, 58, 60, 61, 80, 82, 102, 119, 120, 136, 142, 150, 152, 155, 156].

6.4.4 Other. Some analyzed papers have proposed frameworks that orchestrate services on a configuration of layers that are different from the ones presented previously. These variations are summarized in the following paragraphs.

With the focus on industrial devices and using seamless computing - a standardized service execution platform - the authors of [47] have proposed an architecture that orchestrates resources in four layers: Cloud, Datacenter (on-premises infrastructure), Fog and IoT. In Lamen [118], the

orchestrator is located on the cloud, but the resources to execute the services are in the IoT layer. The framework considers that the devices in the IoT layer will form a geo-location based cluster and one of them will be elected as the leader, named as the Mediator. The Mediator will manage the other devices to execute the services requested by the orchestrator. In Sunstone [44], the orchestration also targets the Cloud and Fog layers, but the architecture has a discovery mechanism that operates on Internet-scale, allowing the fog layer to be distributed across several independent sites. The approach uses internet protocols (i.e. BGP and DNS) to discover fog sites. After discovery, their resources can be shared with applications and services orchestrated by Sunstone. The Maestro framework [145] works only in the IoT layer. One of the IoT devices is defined as the broker and it has the role of orchestrating the tasks that compose a requested service so they can be executed on the resources of one or more proximal devices.

Several proposals (19 of 50) have included the IoT layer as a place where their orchestration proposals could work on. But as this layer could represent the end user devices, resource management can be difficult since it raises privacy issues and potentially mobility management. These issues were not addressed by the majority of works. In use cases where IoT Layer is related to the end users and Cloud and Fog layers are related to the service provider, the utilization of IoT devices as execution environments might not be feasible without clear cost and operation models.

6.5 Functionalities of a Generic Fog Computing Orchestration Architecture

Aiming to answer RQ5, the description and architecture of functional modules proposed to address fog orchestration were extracted from the analyzed papers. Some functionalities are well known and easily recognizable as being part of an orchestration framework (e.g. resource management). They have appeared in the majority of papers as can be seen later in this section. But other functionalities, such as a function that strategically (re)locate mobile fog nodes to better deal with local changes on computation load, are very specific and may not be well known.

To expose the approaches and solutions given by the literature, this paper presents a generic architecture that consolidates them. The architecture is presented in Figure 11. It shows all the orchestration's functionalities that were described in the analyzed papers. The following subsections will describe these functionalities, the way they interact with each other, and point out the papers that have proposed them.

6.5.1 Admission Control of Incoming Requests. An orchestration framework must have a way of receiving the requests to provide the services available within the fog computing provider's portfolio. This functionality may verify a requester's credentials before serving him, being an additional step for privacy preserving and checking if the requester has a proper authorization to run the requested service.

Although the majority of analyzed papers have not addressed this interface and considered that the requests will arrive in some way that is not described, few of them addressed this functionality and provided an interface to admit requests for service delivery and execution [3, 34, 134, 156]. Some proposals have defined and published an **Application Programming Interface (API)**. This API can be integrated to legacy request management systems or be instantiated directly in order to access the service requested [23, 52, 59, 119].

The admission control of incoming request module can also help to guarantee SLAs and **Quality of Experience (QoE)** agreements. After receiving a request, it is possible to verify if the fog infrastructure has available resources to provide and also if the requesting user has the minimum resources required (e.g. signal strength, battery power) to wait, interact and timely receive the results. A request can be rejected by this functionality when it is not possible to fulfill it [104].

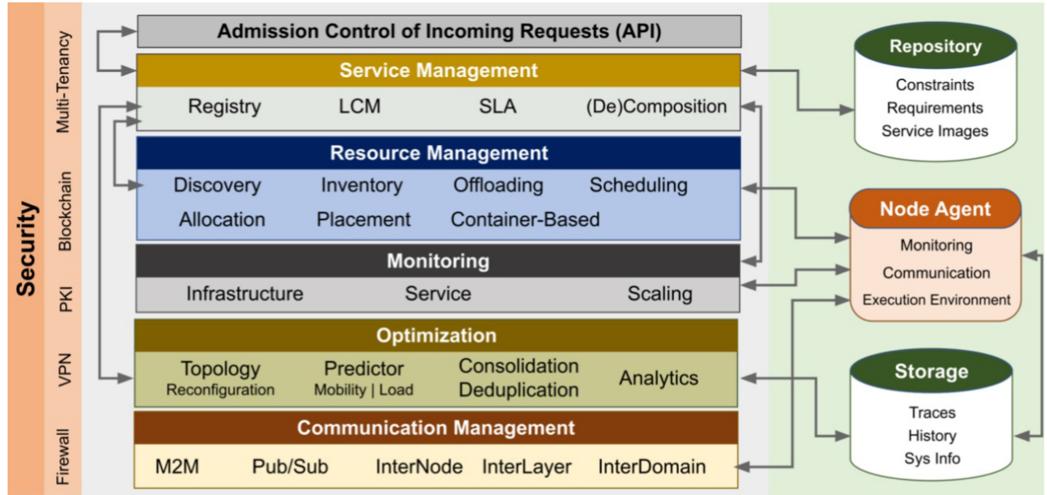


Fig. 11. A generic fog computing orchestration architecture.

6.5.2 Service Management. A service is an executable piece of code that does some kind of processing and delivers a result [105]. **Service Management (SM)** is the functionality that deals with the life cycle of a service within a provider's portfolio.

Several analyzed papers have proposed SM module in their architectures [9, 16, 61, 87, 95, 102, 110, 111, 120, 125, 136, 148]. To do what it was made to do, a service will need a set of resources, dependencies that must be provided prior to its execution. Besides, a service may have specific requirements to function properly and achieve its objectives, e.g. a minimum amount of execution memory. This information must be provided at the time of service registry and it can be used in almost every life cycle phase after a service request has arrived. The executable code, or service image, must also be provided, following the rules defined by the orchestration framework about the supported executing environments. Executable code and related information about the constraints and requirements form a database called service repository, proposed by the papers [3, 5, 16, 24, 44, 52, 69, 82, 99, 118, 119, 134, 136, 141, 152, 156].

Service management may decompose a monolithic service into micro-services to execute them independently, guaranteeing isolation between them and providing a better use of resources [95]. Service management is also responsible for requesting a resource increase when monitoring information about a service points to a risk of not fulfilling a given requirement. This resource increase will be analyzed by the resource management module and result in one of the following situations:

- (1) Fog node executing the service has available resources: **Resource Management (RM)** requests the improvement;
- (2) Fog node executing the service does not have available resources: RM requests service offload (migration) to a resource-richer node or to the cloud;
- (3) Fog node executing the service does not have available resources and there is not another node that can execute the service properly: SM will be informed and will decide whether a lower execution performance can be accepted or not.

Service management interacts with resource management to check the availability of fog infrastructure at specific moments and to communicate a change in the needed resources of each service.

6.5.3 Resource Management. Fog computing is a dynamic environment and its resources, i.e. hardware and software components, must be well managed to cope with the agreements made with the users. Resource management in fog computing is a complex subject that aims to address the life cycle of the portion of the shared infrastructure that is available to support the services provided by the fog operator [55]. Resource management is the most frequently provided functionality among analyzed papers. It is also the most-cited goal of an orchestration framework, as shown in the Section 6.1.

Fog orchestration frameworks use resource management to provide the environment where services will be executed, helping to reach the purpose of extending the cloud to the edge and to comply with agreed SLAs. Resource management can be divided on several sub processes (e.g. resource discovery, allocation, provisioning, scheduling, placement, etc.) [75], [45], according to the phase of resource life cycle management involved. There is no consensus in the literature about these sub processes' names and sequence and it is not the intention of this paper to get deeper into this issue. None of the analyzed papers presented functionality that has covered all the sub processes of resource management.

This subsection approaches the focus given by analyzed papers on this subject. The objective is twofold. First, it presents a view from a higher level of abstraction, from the point of view of an orchestration framework. Second, it exposes the details of some specific proposals, bringing them to light to help further studies in this specific area.

The resource discovery sub process is responsible for finding out new fog nodes that could integrate with the fog infrastructure. To be found and integrated, fog nodes can use self-announcement as an active strategy to send the information about their availability and resources to the orchestration framework, usually located on the same domain. To find fog nodes located on other domains, the orchestrator can search for them through the Internet. As soon they are found, a previously established protocol is initiated to confirm that is safe to integrate them. In ExEC [159], after receiving requests that come from other domains, the orchestrator can automatically find an independent fog provider with the use of already proven Internet protocols and tools (e.g. DNS, traceroute) and propose a commercial agreement with the use of blockchain. After the agreement is established, the discovered fog nodes are integrated to the fog infrastructure. A similar Internet-scale discovery strategy is used by Sunstone [44]. The following papers have approached resource discovery in their orchestration framework: [20, 23, 24, 58, 61, 84, 87, 99, 110, 111, 120, 156, 159].

The information about new fog nodes and the status of availability of their resources is recorded on a repository (a resource inventory) that is managed by the orchestration framework in place. This repository can be updated by periodic messages sent by the nodes, by monitoring events and by service management actions. The following papers have described a resource inventory: [3, 15, 23, 24, 28, 44, 82, 87, 99, 111, 119, 156].

After a request is accepted by the orchestrator, resource management must allocate the resources needed to run the requested service. According to Kalyvianaki [63], resource allocation is the task that aims to provide, in an appropriate manner, the computational resources so that the service or application can achieve its defined performance and QoS goals. This subject is widely researched in the area of computing since computational resources are often limited and, therefore, must be well used. In recent years, several papers have been published on this subject in the most varied computational paradigms, such as Grid Computing [41], **High Performance Computing (HPC)** [48], and Cloud Computing [124]. The resource allocation is relevant in several areas of research because it aims at the optimized use of available resources [64]. Resource allocation, also referred as resource provisioning [68], is a sub process of RM responsible for selecting and reserving, from the resource inventory, the best available resources to execute the service. In the paper [34], to avoid problems related to over and under-allocation of resources, a deep learning approach is used to

make decisions about scaling the fog resources. In TACRM [21], a trust value is estimated for each user. Based on this value, a user's priority is defined and it is used to define the amount of resources allocated. Users with highest priorities should receive the maximum of requested resources, if available. The following papers have proposed resource allocation/provisioning functionality: [9, 15, 21, 23, 24, 34, 52, 61, 95, 102, 125, 145, 148].

Resource scheduling in fog computing is responsible for reaching the best viable assignment of available resources to service requirements, according to the scheduling policies in place. Its goals are both to meet QoS requirements and to lessen the execution time for the services [45]. This RM sub-process was described by [12, 28, 52, 60, 61, 95, 118, 119, 145]. The placement of services makes use of previous RM described sub-processes to find the best resource set that could execute all the service's components and meet QoS and SLAs [117]. It was implemented by the proposals in [15, 16, 28, 32, 44, 80, 82, 84, 95, 99, 118, 132, 141, 147, 148, 150, 156, 158]. Some papers have focused on the computation offloading, a function that puts on a resource-richer node (or in the cloud) a service (or part of it), preventing the requester from exhausting its resources, and sending back the results. The following papers have addressed computation offloading in fog computing orchestration: [15, 20, 58, 60, 82, 104, 120].

After a service has started its execution, the scenario can change dynamically and it could be necessary to take actions to guarantee a desirable performance and optimize the resource usage. Auto-scaling is the resource management sub process that is in charge of taking these actions [128]. Various approaches were proposed in the literature to address this problem. For instance, horizontal scaling [94], adding or removing service instances according to the load increase or decrease. Or vertical scaling [38], resizing the resources allocated to the service. The scaling behavior can be proactive, based on techniques that try to predict the load variation and act to anticipate it, or it can be reactive, based on predefined rules that take action after load variation occurs and reach specific thresholds, e.g. CPU usage above 90% [128]. In RECAP [102], the framework proposed performs application modeling, recording application characteristics, dependencies and limitations. This information will be used thereafter to autonomously scale the application vertically and horizontally, as needed. The following papers also proposed auto-scaling implementation: [15, 32, 34, 52, 61, 148].

All the analyzed work cited previously in this subsection have proposed resource management functionalities designed to achieve the specific goals they intended to. PiCasso [69], a service orchestration framework targeted to **Small Board Computer (SBC)** as underlying infrastructure, utilized containers as an execution environment. The following papers also rely on containers as execution environment: [5, 16, 95, 119, 148, 156].

Other papers have used modified container orchestration solutions to deal with resource management inside their proposals. The authors of [152] have extended Kubernetes [67], adding a local service image repository and a labeling system, used to map service requirements to fog nodes, considering the nodes' resources and location, explicitly recorded on labels attached to them on Kubernetes. The papers [12, 47, 119] also used Kubernetes as part of their resource management strategy.

6.5.4 Monitoring. Fog is a paradigm that aims to provide computational capacity in the vicinity of the user by means of fog nodes, heterogeneous resource-restricted devices that are linked to the user device by one or a few hops through a local network (wired or wireless) [11]. These characteristics bring uncertainty about resource availability. The resources must be monitored frequently to guarantee an updated status of the infrastructure that will be used to provide the services. Besides verifying availability, monitoring is also about the measurements needed to analyze the requirements that a resource (or service) must attend to, according to the SLAs or QoEs. Thus, monitoring is also a function that is of primary importance in a fog computing orchestration framework.

Due to aforementioned restrictions of fog nodes, there has to be a local monitoring function of its resources (energy, CPU, memory, storage, accelerators, attached sensors). This information can be used to make decisions about accepting new services, about offloading, etc. According to fog computing topology of control in place, as discussed in Section 6.3, this information can be stored locally to support autonomous decision making, exchanged with other nodes or, to support a centralized decision-making, exchanged to a central module to give a broader view of all the infrastructure.

Although this functionality be of primary importance to fog computing orchestration, the proposals barely cited it without offering detailed information about its behavior, complexity and issues. This is an identified research gap that could be explored in future works. The following proposals (40 of 50) have included monitoring in the functionalities they provide:[5, 9, 12, 15, 16, 20, 21, 23, 24, 28, 32, 34, 44, 47, 58–61, 69, 80, 84, 87, 95, 102, 104, 110, 111, 118, 120, 125, 132, 134, 136, 141, 142, 147, 148, 150, 152, 156].

6.5.5 Optimization. According to Bellendorf and Mann [10], “Optimization plays an important role in fog computing research, since the fundamental goals of fog computing are related to optimization of metrics like latency, energy consumption, or resource utilization”. An optimization problem is typically structured by variables that encode decisions to be made, constraints that the variables should satisfy and an objective function. The goal of optimization is to find a solution that minimizes or maximizes the objective function and satisfies the constraints [74]. The optimization module of a fog orchestrator implements algorithms and techniques to minimize some metrics (e.g. latency, network load) and/or maximize others (e.g. resource availability). These optimizations can help the orchestrator to achieve other composite metrics (e.g. QoS, QoE, SLA). This module must also consider the requirements and constraints of each service that are available in the Service Repository, as described in Section 6.5.2.

In the proposal presented by [142], a mobility predictor function is available. The service management functionality uses it to predict the location of a moving user and, with this information, place or replace a service requested by that user and comply with the SLA. This approach is also used by [155].

The paper [147] proposed an optimization function that calculates the optimal re-configuration of topology in terms of location of services to guarantee agreed QoE/QoS. This strategy is also adopted by [95, 150, 158]. The work of [95] uses a Bayesian Iterative Reinforcement Learning approach to reconfigure the topology of services.

Other approaches related to service distribution are (1) Decompose the service [134] into a **Direct Acyclic Graph (DAG)** of micro-services and make a deduplication [145], finding the sub-DAGs that are used by more than one service and preventing their multiplication in the infrastructure and (2) Consolidate the services [158] using the same approach of cloud datacenters’ server consolidation. The goal is to have a minimum number of service replicas and redirect user requests to the consolidated services during the process, optimizing resource usage. The approach proposed by [60] uses analytics on historical data of peer domains to improve the decision of offloading a service among domains. The use of analytics on already collected data is also made by [58].

Predicting a future scenario to anticipate actions is an optimization strategy found in some papers. In [132], each Fog device runs an agent that is responsible for the orchestration in its segment (network range). One device at a time is also responsible for area orchestration, a function that looks at area (sum of all segments) load and (re)locates mobile edge devices to segments that need more resources. The paper [141] also uses network load prediction to optimize its resource and service management actions. Besides predicting network load, communication load prediction [155] and computation load prediction [104, 118] are other approaches found in the literature.

Optimization can be used on QoS verification. In [16], a model-free machine learning technique that fits well on resource restricted devices, called reinforcement learning, is used to accomplish this task.

CHARIOT [111] uses a **domain-specific modeling language (DSML)** to describe the fog environment: fog nodes and service portfolio. It also formulates **Satisfiability Modulo Theories (SMT)** constraints that encode environment properties and requirements, enabling the use of SMT solvers [26] to dynamically compute optimal system (re)configuration at runtime. The use of description languages to model fog nodes and services is also used by [15, 16, 136].

6.5.6 Communication Management. Although communication is obviously extremely important to achieve the goals of an orchestration framework, only a handful of papers have included this functionality in their proposed architecture and offered details about the entities and means of communication they use.

As seen in the previous subsections there is a need for communication and an intense data exchange to implement the decentralized [21] and distributed [9] orchestration control topologies (Section 6.3), to do resource management considering other domains [60] or different fog sites [44].

Communicating with resource-restricted and heterogeneous devices using an unstable network is also a challenge that has to be addressed. These fog computing characteristics will restrict the communication protocols and tools that are proper to use in this scenario. In the paper [24], the authors have proposed a **machine-to-machine (M2M)** communication module that is deployed to the fog nodes for talking to attached devices and sensors on the southbound interface. The module is responsible for collecting generated data and monitoring information (resources availability, battery level etc). The authors have used OpenMTC M2M Framework [146] to implement the proposed module. The paper [120] also used M2M communication. Other approaches were proposed for southbound communication: inter-layer communication [80], **Geographically Addressed (GA) Messaging** [5] and plugin-based [52].

In [142], the authors described a module that controls the communication among different orchestrator instances that may be deployed simultaneously in the three layers: IoT, Fog and Cloud. The paper [60] proposed a communication management functionality responsible for the westbound communications, i.e. frequent exchanged messages between peer fog domains. The messages exchanged have a threefold objective: (1) Communicating available resources and rules to use them; (2) Measuring transmission delays that will be used to guarantee QoS requirement of IoT services, and (3) Creating a database of historical data of each domain to improve offload decisions by the use of analytics. The paper [120] also utilized peer-to-peer communication on the westbound interface. The work of [58] implemented this communication by UDP broadcast while the proposal of [20] utilizes a shared memory approach.

The paper [15] used communication management for two reasons: (1) in an internal module, to implement a distributed consensus algorithm (DRAGON) that decides which resources will be allocated for each service; (2) in an external module, to deliver a message bus, responsible for the propagation of changes about services configuration and infrastructure status. The Ecco framework [20] also exchanges data in two ways: (1) intra-node, when the data transfer happens between two services deployed on the same node or between a service and the central orchestrator (called Maestro); and (2) inter-node, when the transfer takes place between two services on different nodes. In the last case, a shared memory approach was proposed, implemented by a custom module called **memory manager (MM)**.

Regarding the communication model in use, **publisher/subscriber (pub/sub)** was proposed by [15]. The papers [28, 156] implemented pub/sub with the use of MQTT [89], a lightweight messaging protocol for IoT, used also by [110] on the northbound interface. The paper [16] has

used NGSI [96], a context management protocol. Message queues were adopted by [5, 16] with the use of RabbitMQ tool [143], and by [111], with the use of ZeroMQ tool [54]. The paper [136] used NETCONF [33], a standard **Internet Engineering Task Force (IETF)** protocol, to distribute and update device configuration on fog computing infrastructure. In the proposal presented by [84], the **Web Application Messaging Protocol (WAMP)** is used for node-related interactions and to bridge the communication with other components and boards.

6.5.7 Node Agent. To run services on fog nodes it is necessary to have an execution environment deployed on them. Besides, to orchestrate services among different fog nodes, it is important to know precisely the resource availability. These two operations can be executed remotely. The former can be executed using a container orchestration solution on the nodes. The latter can be implemented by calling operating system primitives remotely, but in case of network unavailability the information cannot be recovered and can be lost. In both cases, the approaches demand the environment and configuration of fog nodes to comply with each one individually.

An approach that can be more flexible is to deploy a Fog Node agent, a module that is responsible to manage the node locally. This module can be implemented on different platforms, can do local operations, such as saving monitoring information and managing the life cycle of an executing service, and can communicate with other agents or with a central orchestration module, depending on the control topology of the orchestration framework in place. The following papers have described node agents with specific functions: [5, 15, 16, 20, 21, 23, 24, 32, 52, 69, 80, 84, 99, 110, 111, 120, 125, 132, 145, 148, 156].

6.5.8 Security. Information security and privacy are important subjects in a fog computing environment. The hardware limitations of some fog nodes and attached sensors and actuators, distributed architecture and network uncertainty are fog characteristics that increase the attack surface and limit the security solutions that could be applied [115].

Regarding the use of security on data communication and storage, only a few articles have established the conditions for implementation [24, 142]. Although their architecture described security modules, there were no implementations yet and their proposals comprises only a conceptual architecture.

Security solutions already proven in other environments were also used. Proposals made by [5, 28] used VPN to secure the communications. The authors of [148] proposed a firewall to improve the communication security and in [61] and [110], the authors proposed the use of a **public key infrastructure (PKI)** to encrypt the communications between the orchestration framework and the service requesters. The paper [103] has used blockchain, a secure distributed, replicated and shared data structure [17], to implement a trusted orchestration that guarantees data privacy and provenance inside the orchestration functionalities.

Fog computing extends the cloud to the edge in a three-layered architecture as shown in Figure 1. According to specific scenarios, e.g. as industrial implementations, some fog nodes can be resource rich and support execution of services from more than one user, which is known as multi-tenancy. The papers [52], [136] and [158] have considered multi-tenancy on the orchestration architecture proposed.

6.5.9 Summary of Results. To summarize the results and to allow researchers easily find the characteristics of each analyzed proposal regarding the research questions, Table 3 presents all results by paper. The papers are ordered by year of publication. To show RQ1 answers, we used goal codes defined in Table 2. To present the answers to RQ5, we have shortened the functionality's name as follows: **AC-Admission Control of incoming requests; SM-Service**

Table 3. Analyzed Papers' Answers to Research Questions

Paper	Year	Goal (RQ1)	Entity (RQ2)	Topology (RQ3)	Layer (RQ4)	Functionalities (RQ5)								
						AC	SM	RM	MO	OP	CM	NA	SE	
[52]	2015	RM	Application	Centralized	Fog/Cloud	✓		✓			✓	✓	✓	
[145]	2016	SLA	Task	Centralized	IoT			✓		✓		✓	✓	
[125]	2016	RM	Task	Centralized	Fog/Cloud		✓	✓	✓		✓	✓		
[5]	2016		Application	Centralized	Fog/Cloud		✓	✓			✓	✓	✓	
[102]	2017	SLA	Application	Centralized	All		✓	✓	✓	✓				
[24]	2017		Service	Decentralized	Fog		✓	✓	✓		✓	✓	✓	
[142]	2017	SLA	Service	Decentralized	All		✓	✓	✓	✓			✓	
[69]	2017		Application	Centralized	Fog		✓	✓					✓	
[119]	2017	RM	Application	Centralized	All		✓							
[155]	2017	RM, SLA	Moving Query	Decentralized	All						✓			
[118]	2017	RM	Service	Decentralized	IoT/Cloud	✓		✓	✓	✓				
[136]	2017	NFV-5G	Service	Centralized	All		✓	✓	✓		✓		✓	
[141]	2017	SLA, APP	Service	Centralized	Fog/Cloud		✓	✓	✓					
[150]	2017		Application	Centralized	All			✓	✓					
[148]	2017	RM	Application	Decentralized	Fog/Cloud		✓	✓	✓			✓	✓	
[156]	2017	SLA	Application	Centralized	All		✓	✓	✓		✓	✓		
[9]	2018	RM	Task	Distributed	All		✓	✓	✓		✓			
[147]	2018	SLA, ENE	Service	Centralized	Fog		✓	✓	✓					
[60]	2018		Service	Distributed	All		✓	✓	✓		✓			
[152]	2018		Application	Centralized	All		✓	✓						
[103]	2018	SEC	Service	Centralized	Fog		✓						✓	
[16]	2018	RM	Task	Centralized	Fog/Cloud	✓	✓	✓	✓	✓	✓	✓	✓	
[158]	2018	RM	Application	Centralized	Fog		✓		✓				✓	
[120]	2018		Application	Centralized	All		✓	✓	✓		✓	✓	✓	
[111]	2018	APP	Service	Centralized	Fog	✓	✓	✓	✓	✓	✓	✓	✓	
[21]	2019	RM	Task	Decentralized	Fog/Cloud		✓	✓			✓	✓	✓	
[159]	2019		Service	Centralized	Fog/Cloud		✓		✓		✓	✓	✓	
[15]	2019		Service	Distributed	All		✓	✓			✓	✓		
[132]	2019		Task/Hardware	Distributed	Fog/Cloud		✓	✓	✓				✓	
[32]	2019	RM	Service	Centralized	Fog		✓	✓					✓	
[12]	2019		Service	Centralized	Fog/Cloud		✓	✓						
[58]	2019	SLA, ENE	Task	Distributed	All		✓	✓	✓					
[28]	2019	SLM	Analytical Pipel.	Centralized	All		✓	✓			✓		✓	
[3]	2019		Task	Decentralized	Fog	✓		✓						
[95]	2019	SLA	Service	Centralized	Fog/Cloud		✓	✓	✓					
[99]	2019		Application	Centralized	Fog		✓				✓	✓		
[59]	2019	SLA	Task	Centralized	Fog/Cloud	✓	✓	✓						
[84]	2019			Centralized	Fog/Cloud		✓	✓			✓	✓		
[20]	2020	SLM	Edge Function	Centralized	Fog		✓	✓			✓	✓		
[23]	2020		Service	Centralized	Fog	✓		✓	✓				✓	
[104]	2020		Service	Centralized	Fog		✓	✓	✓					
[47]	2020	SLM	Application	Centralized	All		✓	✓						
[44]	2020	RM	Application	Decentralized	Fog/Cloud		✓	✓						
[80]	2020	RM	Service	Centralized	All		✓	✓			✓	✓	✓	
[82]	2020		Task	Centralized	All		✓							
[134]	2020		Service	Centralized	Fog/Cloud	✓			✓	✓				
[61]	2020		Application	Decentralized	Fog		✓	✓	✓				✓	
[87]	2020		Application	Centralized	Fog/Cloud		✓	✓	✓				✓	
[34]	2021	COS	Application	Centralized	Fog/Cloud	✓		✓	✓	✓				
[110]	2021		Task	Distributed	Fog/Cloud		✓	✓	✓	✓	✓	✓	✓	
						Count	9	12	48	40	18	22	21	17

Management; RM-Resource Management; MO-Monitoring; OP-Optimization; CM-Communication Management; NA-Node Agent and SE-Security.

By analyzing Table 3 it is possible to see that none of the papers have implemented all functionalities presented in this Section. The generic fog computing orchestration architecture presented in Figure 11 consolidated all analyzed proposals. It shows the orchestration functionalities addressed

and also the choices made by the authors to implement them, being helpful to researchers that want to know better the state of the art on this subject.

As a recent area of research, the proposals have first tried to aggregate, in a structured and coordinated way, the several functionalities related to orchestration in fog, focusing on the interactions among them and considering cross-layer communications. Thus, resource management and monitoring were the functionalities that are most implemented by the analyzed proposals, because they are directly related to services' execution environment and are fundamental for the implementation of some of the essential fog characteristics, mentioned in Section 2, such as low latency (achieved through providing resources close to the user, and measured and accompanied by monitoring), real-time interactions and scalability, which is leveraged by monitoring events timely indicating the need for change on resource allocation.

There are many fog computing papers that have focused on each functionality independently, e.g. Optimization [10], Security [90], getting a broader and deeper view of the subject and analyzing different approaches. The orchestration research could focus on the management and coordination issue of how to get all the functionalities working properly simultaneously, all of them contributing to specific goals that the orchestration framework in place has proposed to achieve. In this sense, the use of policies to determine the choice of optimization algorithms and techniques that must took place in each of integrated functionality may be a research direction. Thus, instead of proposing something new about one or two functionalities, a fog orchestration proposal could adopt the state of the art of each one, adapting it to follow the policy stated in the orchestration framework by means of parameterization.

7 CHALLENGES

This work analyzed 50 papers related to orchestration in fog computing. The five research questions were answered with the results found on the analysis of the papers, which brought to light the issues addressed by the authors and the proposals they offered to cope with them. The analysis also made it possible to identify open challenges that need more research and proposals in the context of this work. They are privacy and security, evaluation in real environments, and a standardized execution environment. The intention is not to be an exhaustive list, but that the information presented below can support future work on the issue of orchestration in fog computing.

7.1 Privacy and Security

Privacy and information security are challenges that need more attention from academia in the context of orchestration in fog computing. From the 50 studies analyzed in this article, privacy and security in orchestration were considered by only 17 of them (34%). None of the analyzed papers have implemented authentication and authorization as a way to verify the identity of service requester and to preserve data privacy [78].

Considering the strategies used by an orchestration framework, it is possible that re-allocations, replication and migration of services may be necessary to cope with user mobility, fault handling and to ensure adequate service availability and performance [142, 150]. Considering the number of hops the data travels, or the quantity of devices it is copied or shared to, the risk of theft or misuse may be high [2]. Using the Cloud layer when there is a lack of resources in the Fog layer is a scenario implemented by many proposals, as shown in Section 6.4. A federation of fog computing domains could be a better alternative, since it can warrant fog computing benefits, e.g. lower latency. On the other hand, authentication and authorization have to be enforced to increase data privacy. Only two of the analyzed papers have proposed a federation of fog environments, [16] and [158], and this scenario deserves more attention in future research.

7.2 Evaluation in Real Environments

Orchestration in fog computing is a recent area of study as shown in Figure 2. Most of the analyzed papers have proposed an architecture for orchestration and described the interactions among its modules. But only a few of them have used simulators (13 of 50 papers) to evaluate and compare their proposals with alternative solutions. A smaller number of papers (12 of 50) have evaluated their proposals on real testbeds.

Using simulators, there is no way to create a database with the history of service execution on real fog nodes. These databases can be used jointly with artificial intelligence to allow the development of predictive algorithms [8, 140] to be used in the optimization module of an orchestration framework as seen in the Section 6.5. Therefore, there are only a few works in the literature that have evaluated their fog computing orchestration proposals in real implementation environments.

7.3 Standardized Execution Environment

A federation of public and private providers, connected through the Internet, is largely adopted in cloud computing [25, 116], expanding the resource availability with the increase of management complexity. In the fog computing scenario there is a lack of studies about this topic. Due to heterogeneity of devices that compose a fog environment, it is important also to research about the use of standardized execution environments that support service execution seamlessly through the federated fog providers.

8 CONCLUSION

This article provided, through a systematic literature review, an overview of the state of research on the process of orchestration in fog computing, in addition to presenting the characteristics of this computational paradigm. To achieve this goal, it was necessary first to define the scope of the term orchestration in fog computing, since there is no consensus about this in the literature. Thus, the mapping process continued with the definition of five research questions that guided the search, the selection criteria and the evaluation of the publications found.

Evaluating the 50 selected publications, it was possible to answer all the research questions, create a generic fog computing orchestration architecture, and present some challenges that are still not well addressed in the literature, concluding that there are still many questions that need to be investigated by academia. Thus, by presenting a systematic review of the literature specifically on the orchestration for fog computing, the present work contributes significantly to the fog computing knowledge base, providing support to researchers to direct their future works to the existing gaps.

ACKNOWLEDGMENTS

The authors would like to thank CAPES, a Brazilian institution that funds research and publishing, for facilitating the access to many publications through CAPES' journal portal (<https://www.periodicos.capes.gov.br>).

REFERENCES

- [1] Mohammad Aazam and Eui Nam Huh. 2015. Fog computing micro datacenter based dynamic resource estimation and pricing model for IoT. *Proceedings - International Conference on Advanced Information Networking and Applications, AINA 2015-April*, March (2015), 687–694.
- [2] Mohammad Aazam, Sheralli Zeadally, and Khaled A. Harras. 2018. Offloading in fog computing for IoT: Review, enabling technologies, and research opportunities. *Future Generation Computer Systems* 87 (2018), 278–289.
- [3] N. Agrawal, J. Rellermeier, and A. Y. Ding. 2019. IoT resource-aware orchestration framework for edge computing. In *CoNEXT 2019 Companion - Proceedings of the 15th International Conference on Emerging Networking EXperiments and Technologies*. 62–64. <https://doi.org/10.1145/3360468.3368179>

- [4] F. Al-Doghman, Z. Chaczko, A. R. Ajayan, and R. Klempous. 2016. A review on fog computing technology. In *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. 001525–001530. <https://doi.org/10.1109/SMC.2016.7844455>
- [5] Brian Amento, Bharath Balasubramanian, Robert J. Hall, Kaustubh Joshi, Gueyoung Jung, and K. Hal Purdy. 2016. FocusStack: Orchestrating edge clouds using location-based focus of attention. In *2016 IEEE/ACM Symposium on Edge Computing (SEC)*. IEEE, 179–191. <https://doi.org/10.1109/SEC.2016.22>
- [6] John L. Anderson. 1983. Autonomous systems intelligence. In *Proceedings of the 1983 Annual Conference on Computers: Extending the Human Resource*. 229–233.
- [7] Hamid Reza Arkian, Abolfazl Diyanat, and Atefe Pourkhalili. 2017. MIST: Fog-based data analytics scheme with cost-efficient resource provisioning for IoT crowdsensing applications. *Journal of Network and Computer Applications* 82 (2017), 152–165.
- [8] A. Asensio, X. Masip-Bruin, R.J. Durán, I. de Miguel, G. Ren, S. Daijavad, and A. Jukan. 2020. Designing an efficient clustering strategy for combined fog-to-cloud scenarios. *Future Generation Computer Systems* (2020).
- [9] Cosmin Avasalcai and Schahram Dustdar. 2018. Latency-aware decentralized resource management for IoT applications. In *Proceedings of the 8th International Conference on the Internet of Things*. 1–4.
- [10] Julian Bellendorf and Zoltán Ádám Mann. 2020. Classification of optimization problems in fog computing. *Future Generation Computer Systems* 107 (2020), 158–176.
- [11] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. 2012. Fog computing and its role in the Internet of Things. In *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing (MCC’12)*. ACM, New York, NY. 13–16. <https://doi.org/10.1145/2342509.2342513>
- [12] A. Buzachis, A. Galletta, A. Celesti, L. Carnevale, and M. Villari. 2019. Towards osmotic computing: A blue-green strategy for the fast re-deployment of microservices. In *2019 IEEE Symposium on Computers and Communications (ISCC)*, Vol. 2019-June. 1–6. <https://doi.org/10.1109/ISCC47284.2019.8969621>
- [13] Cambridge English Dictionary. ORCHESTRATION | meaning. 2021. Retrieved Feb 17, 2021 from <https://dictionary.cambridge.org/dictionary/english/orchestration>.
- [14] Andrew Campbell, Geoff Coulson, Francisco Garcia, and David Hutchison. 1992. A continuous media transport and orchestration service. In *Conference Proceedings on Communications Architectures & Protocols*. 99–110.
- [15] Gabriele Castellano, Flavio Esposito, and Fulvio Risso. 2019. A service-defined approach for orchestration of heterogeneous applications in cloud/edge platforms. *IEEE Transactions on Network and Service Management* 16, 4 (2019), 1404–1418. <https://doi.org/10.1109/TNSM.2019.2941639>
- [16] B. Cheng, E. Kovacs, A. Kitazawa, K. Terasawa, T. Hada, and M. Takeuchi. 2018. FogFlow: Orchestrating IoT services over cloud and edges. *NEC Technical Journal* 13, 1 (2018), 48–53.
- [17] Konstantinos Christidis and Michael Devetsikiotis. 2016. Blockchains and smart contracts for the Internet of Things. *IEEE Access* 4 (2016), 2292–2303.
- [18] Cisco. Fog Computing and the Internet of Things: Extend the Cloud to Where the Things Are. 2021. Retrieved Feb 17, 2021 from <https://www.cisco.com/c/dam/en-us/solutions/trends/iot/docs/computing-overview.pdf>.
- [19] Breno GS Costa, Marco Antonio Sousa Reis, Aletéia PF Araújo, and Priscila Solis. 2018. Performance and cost analysis between on-demand and preemptive virtual machines. In *CLOSER*. 169–178.
- [20] V. Cozzolino, J. Ott, A. Y. Ding, and R. Mortier. 2020. ECCO: Edge-cloud chaining and orchestration framework for road context assessment. In *Proceedings - 5th ACM/IEEE Conference on Internet of Things Design and Implementation, IoTDI 2020*. 223–230. <https://doi.org/10.1109/IoTDI49375.2020.00029>
- [21] Wided Ben Daoud, Mohammad S. Obaidat, Amel Meddeb-Makhlof, Faouzi Zarai, and Kuei-Fang Hsiao. 2019. TACRM: Trust access control and resource management mechanism in fog computing. *Human-centric Computing and Information Sciences* 9, 1 (2019), 1–18.
- [22] Amir Vahid Dastjerdi, Harshit Gupta, Rodrigo Neves Calheiros, Soumya K. Ghosh, and Rajkumar Buyya. 2016. Fog computing: Principles, architectures, and applications. *CoRR* abs/1601.02752 (2016).
- [23] G. Davoli, D. Borsatti, D. Tarchi, and W. Cerroni. 2020. FORCH: An orchestrator for fog computing service deployment. In *2020 IFIP Networking Conference (Networking)*. 677–678.
- [24] Mathias Santos de Brito, Saiful Hoque, Thomas Magedanz, Ronald Steinke, Alexander Willner, Daniel Nehls, Oliver Keils, and Florian Schreiner. 2017. A service orchestration architecture for fog-enabled infrastructures. IEEE, 345 E 47th Street, New York, NY, 127–132.
- [25] Leonardo Rebouças de Carvalho and Aleteia Patricia Favacho de Araujo. 2020. Performance comparison of terraform and cloudfy as multicloud orchestrators. In *2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID)*. IEEE, 380–389.
- [26] Leonardo De Moura and Nikolaj Bjørner. 2008. Z3: An efficient SMT solver. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 337–340.

- [27] Nathan F. Saraiva de Sousa, Danny A. Lachos Perez, Raphael V. Rosa, Mateus A.S. Santos, and Christian Esteve Rothenberg. 2019. Network service orchestration: A survey. *Computer Communications* 142 (2019), 69–94.
- [28] J. Díaz-De-Arcaya, R. Miñon, and A. I. Torre-Bastida. 2019. Towards an architecture for big data analytics leveraging edge/fog paradigms. In *ACM International Conference Proceeding Series*, Vol. 2. 173–176. <https://doi.org/10.1145/3344948.3344987>
- [29] Docker. Docker - Empowering App Development for Developers. 2021. Retrieved Feb 17, 2021 from <https://www.docker.com/>.
- [30] Docker Swarm. Swarm mode overview | Docker Documentation. 2021. Retrieved Feb 17, 2021 from <https://docs.docker.com/engine/swarm/>.
- [31] Koustabh Dolui and Soumya Kanti Datta. 2017. Comparison of edge computing implementations: Fog computing, cloudlet and mobile edge computing. *GloTS 2017 - Global Internet of Things Summit, Proceedings* (2017).
- [32] Bruno Donassolo, Ilhem Fajjari, Arnaud Legrand, and Panayotis Mertikopoulos. 2019. Fog based framework for IoT service provisioning. In *2019 16th IEEE Annual Consumer Communications and Networking Conference (CCNC)*. 345 E. 47th Street, New York, NY. 1–6. <https://doi.org/10.1109/CCNC.2019.8651835>
- [33] Rob Enns, Martin Bjorklund, and Juergen Schoenwaelder. 2006. *NETCONF Configuration Protocol*. Technical Report. RFC 4741, December.
- [34] Masoumeh Etemadi, Mostafa Ghobaei-Arani, and Ali Shahidinejad. 2021. A cost-efficient auto-scaling mechanism for IoT applications in fog computing environment: A deep learning-based approach. *Cluster Computing* (2021), 1–16.
- [35] Ali Fahs and Guillaume Pierre. 2019. Proximity-aware traffic routing in distributed fog computing platforms.
- [36] Chih Tien Fan, Zong You Wu, Che Pin Chang, and Shyan Ming Yuan. 2017. Web resource cacheable edge device in fog computing. *Proceedings - 15th International Symposium on Parallel and Distributed Computing, ISPDC 2016 November 2010* (2017), 432–439.
- [37] Yaoling Fan, Qiliang Zhu, and Yang Liu. 2018. Cloud/fog computing system architecture and key technologies for south-north water transfer project safety. *Wireless Communications and Mobile Computing* 2018 (2018).
- [38] Soodeh Farokhi, Pooyan Jamshidi, Drazen Lucanin, and Ivona Brandic. 2015. Performance-based vertical memory elasticity. In *2015 IEEE International Conference on Autonomic Computing*. IEEE, 151–152.
- [39] Nicolas Ferry, Phu Nguyen, Hui Song, Pierre-Emmanuel Novac, Stéphane Lavirotte, Jean-Yves Tigli, and Arnor Solberg. 2019. Genesis: Continuous orchestration and deployment of smart IoT systems. In *2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)*, Vol. 1. IEEE, 870–875.
- [40] A. Galis, H. Abramowicz, M. Brunner, D. Raz, P. Chemouil, J. Butler, C. Polychronopoulos, S. Clayman, H. de Meer, T. Coupagey, et al. 2009. Management and service-aware networking architectures for future internet position paper: System functions, capabilities and requirements. In *IEEE 2009 Fourth International Conference on Communications and Networking in China (ChinaCom09)*.
- [41] Aram Galstyan, Karl Czajkowski, and Kristina Lerman. 2004. Resource allocation in the grid using reinforcement learning. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 3*. IEEE Computer Society, 1314–1315.
- [42] Jerry Gao, Volker Gruhn, Jingsha He, George Roussos, Wei-Tek Tsai, et al. 2013. Mobile cloud computing research-issues, challenges and needs. In *2013 IEEE Seventh International Symposium on Service-Oriented System Engineering*. IEEE, 442–453.
- [43] Gartner. “Gartner Says the Internet of Things Installed Base Will Grow to 26 Billion Units by 2020”. 2015. Retrieved Feb 17, 2021 from <http://www.gartner.com/newsroom/id/2636073>.
- [44] J. Gedeon, S. Zengerle, S. Alles, F. Brandherm, and M. Mühlhäuser. 2020. Sunstone: Navigating the way through the fog. In *2020 IEEE 4th International Conference on Fog and Edge Computing (ICFEC)*. 49–58. <https://doi.org/10.1109/ICFEC50348.2020.00013>
- [45] Mostafa Ghobaei-Arani, Alireza Souri, and Ali A. Rahamanian. 2019. Resource management approaches in fog computing: A comprehensive review. *Journal of Grid Computing* (2019), 1–42.
- [46] Sukpal Singh Gill, Shreshth Tuli, Minxian Xu, Inderpreet Singh, Karan Vijay Singh, Dominic Lindsay, Shikhar Tuli, Daria Smirnova, Manmeet Singh, Udit Jain, Haris Pervaiz, Bhanu Sehgal, Sukhwinder Singh Kaila, Sanjay Misra, Mohammad Sadegh Aslanpour, Harshit Mehta, Vlado Stankovski, and Peter Garraghan. 2019. Transformative effects of IoT, blockchain and Artificial Intelligence on cloud computing: Evolution, vision, trends and open challenges. *Internet of Things* 8, October (2019), 100118.
- [47] Spyridon V. Gogouitis, Harald Mueller, Sreenath Premnadh, Andreas Seitz, and Bernd Bruegge. 2020. Seamless computing in industrial systems using container orchestration. *Future Generation Computer Systems* 109 (2020), 678–688. <https://doi.org/10.1016/j.future.2018.07.033>
- [48] Narasimha Raju Gottumukkala, Chokchai Box Leangsuksun, Narate Taerat, Raja Nassar, and Stephen L. Scott. 2007. Reliability-aware resource allocation in HPC systems. In *2007 IEEE International Conference on Cluster Computing*. IEEE, 312–321.

- [49] Laura Grit, David Irwin, Aydan Yumerefendi, and Jeff Chase. 2006. Virtual machine hosting for networked clusters: Building the foundations for “autonomic” orchestration. In *First International Workshop on Virtualization Technology in Distributed Computing (VTDC 2006)*. IEEE, 7–7.
- [50] Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, and Marimuthu Palaniswami. 2013. Internet of things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems* 29, 7 (2013), 1645–1660.
- [51] Pooyan Habibi, Mohammad Farhoudi, Sepehr Kazemian, Siavash Khorsandi, and Alberto Leon-Garcia. 2020. Fog computing : A comprehensive architectural survey. *IEEE Access* PP (2020), 1.
- [52] Joacim Halen, Stefan Hellkvist, Stephan Baucke, Fetahi Wuhib, and Yagiz Onat Yazir. 2015. Wind: Management and orchestration in the distributed heterogeneous cloud. In *IEEE World Congress on Services*. 39–46. <https://doi.org/10.1109/SERVICES.2015.15>
- [53] Bo Han, Vijay Gopalakrishnan, Lusheng Ji, and Seungjoon Lee. 2015. Network function virtualization: Challenges and opportunities for innovations. *IEEE Communications Magazine* 53, 2 (2015), 90–97.
- [54] Pieter Hintjens. 2013. *ZeroMQ: Messaging for Many Applications*. O'Reilly Media, Inc.
- [55] Cheol-Ho Hong and Blesson Varghese. 2019. Resource management in fog/edge computing: A survey on architectures, infrastructure, and algorithms. *ACM Computing Surveys (CSUR)* 52, 5 (2019), 97.
- [56] IBM. What is fog computing? 2014. Retrieved Feb 17, 2021 from <https://www.ibm.com/blogs/cloud-computing/2014/08/25/fog-computing/>.
- [57] Michaela Iorga, Larry Feldman, Robert Barton, Michael Martin, Nedim Goren, and Charif Mahmoudi. 2018. *The NIST Definition of Fog Computing*. Technical Report. National Institute of Standards and Technology.
- [58] Fatemeh Jalali, Timothy Lynar, Olivia J. Smith, Ramachandra Rao Kolluri, Claire Hardgrove, Nick Waywood, and Frank Suits. 2019. DEFT: Dynamic edge fabric environment seamless and automatic switching among resources at the edge of IoT network and cloud. In *2019 IEEE International Conference on Edge Computing (IEEE EDGE)*. 77–86. <https://doi.org/10.1109/EDGE.2019.00028>
- [59] S. Javaid, N. Javaid, T. Saba, Z. Wadud, A. Rehman, and A. Haseeb. 2019. Intelligent resource allocation in residential buildings using consumer to fog to cloud based framework. *Energies* 12, 5 (2019). <https://doi.org/10.3390/en12050815>
- [60] Yuxuan Jiang, Zhe Huang, and Danny HK Tsang. 2017. Challenges and solutions in fog computing orchestration. *IEEE Network* 32, 3 (2017), 122–129.
- [61] Lara Lorna Jimenez and Olov Schelen. 2020. HYDRA: Decentralized location-aware orchestration of containerized applications. *IEEE Transactions on Cloud Computing* (2020).
- [62] João Bachiega Junior, Marco Antonio Sousa Reis, Aleteia PF de Araujo, and Maristela Holanda. 2017. Cost optimization on public cloud provider for big geospatial data. In *Proceedings of the 7th International Conference on Cloud Computing and Services Science*. 82–90.
- [63] Evangelia Kalyvianaki. 2009. *Resource Provisioning for Virtualized Server Applications*. Technical Report. University of Cambridge, Computer Laboratory.
- [64] Naoki Katoh and Toshihide Ibaraki. 1998. Resource allocation problems. In *Handbook of Combinatorial Optimization*. Springer, 905–1006.
- [65] Barbara Kitchenham, O. Pearl Brereton, David Budgen, Mark Turner, John Bailey, and Stephen Linkman. 2009. Systematic literature reviews in software engineering—a systematic literature review. *Information and Software Technology* 51, 1 (2009), 7–15.
- [66] Diego Kreutz, Fernando MV Ramos, Paulo Esteves Verissimo, Christian Esteve Rothenberg, Siamak Azodolmolky, and Steve Uhlig. 2014. Software-defined networking: A comprehensive survey. *Proc. IEEE* 103, 1 (2014), 14–76.
- [67] Kubernetes. Kubernetes. 2021. Retrieved Feb 17, 2021 from <https://kubernetes.io/>.
- [68] Imen Ben Lahmar and Khouloud Boukadi. 2020. Resource allocation in fog computing: A systematic mapping study. In *2020 Fifth International Conference on Fog and Mobile Edge Computing (FMEC)*. IEEE, 86–93.
- [69] A. Lertsinsrubtavee, A. Ali, C. Molina-Jimenez, A. Sathiaseelan, and J. Crowcroft. 2017. Picasso: A lightweight edge computing platform. In *2017 IEEE 6th International Conference on Cloud Networking (CloudNet)*. 1–7. <https://doi.org/10.1109/CloudNet.2017.8071529>
- [70] Tom H. Luan, Longxiang Gao, Zhi Li, Yang Xiang, Guiyi Wei, and Limin Sun. 2015. Fog computing: Focusing on mobile users at the edge. (2015), 1–11. arXiv:1502.01815 <http://arxiv.org/abs/1502.01815>.
- [71] Md Mahmud and Rajkumar Buyya. 2016. *Fog Computing: A Taxonomy, Survey and Future Directions*.
- [72] Redowan Mahmud, Kotagiri Ramamohanarao, and Rajkumar Buyya. 2020. Application management in fog computing environments : A taxonomy, review and future directions. 1, 1 (2020).
- [73] Ayesha Abdul Majeed, Peter Kilpatrick, Ivor Spence, and Blesson Varghese. 2020. Modelling fog offloading performance. In *2020 IEEE 4th International Conference on Fog and Edge Computing (ICFEC)*. IEEE, 29–38.
- [74] Zoltán Ádám Mann. 2011. *Optimization in Computer Engineering—Theory and Applications*. Scientific Research Publishing, Inc. USA.

- [75] Sunilkumar S. Manvi and Gopal Krishna Shyam. 2014. Resource management for infrastructure as a service (IaaS) in cloud computing: A survey. *Journal of Network and Computer Applications* 41, 1 (2014), 424–440.
- [76] Antonio Manzalini, Diego R. Lopez, Hakon Lonsethagen, Lucian Suciu, Roberto Bifulco, Marie-Paule Odini, Giuseppe Celozzi, Barbara Martini, Fulvio Risso, Jokin Garay, et al. 2017. A unifying operating platform for 5G end-to-end and multi-layer orchestration. In *2017 IEEE Conference on Network Softwarization (NetSoft)*. IEEE, 1–5.
- [77] Marathon. Marathon: A container orchestration platform for Mesos and DC/OS. 2021. Retrieved Feb 17, 2021 from <https://mesosphere.github.io/marathon/>.
- [78] John Paul Martin, A. Kandasamy, K. Chandrasekaran, and Christina Terese Joseph. 2019. Elucidating the challenges for the praxis of fog computing: An aspect-based study. *International Journal of Communication Systems* 32, 7 (2019), e3926.
- [79] X. Masip, E. Marín, J. Garcia, and S. Sánchez. 2020. *Collaborative Mechanism for Hybrid Fog-Cloud Scenarios*. 7–60. <https://doi.org/10.1002/9781119501121.ch2>
- [80] X. Masip-Bruin, E. Marin-Tordera, A. J. Ferrer, A. Salis, J. Kennedy, J. Jensen, A. Jukan, A. Bartoli, R. M. Badia, M. Cankar, and M. E. Bégin. 2020. mF2C: The evolution of cloud computing towards an open and coordinated ecosystem of fogs and clouds. *Lecture Notes in Computer Science* 11997 LNCS (2020), 136–147. https://doi.org/10.1007/978-3-030-48340-1_11
- [81] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. 2008. OpenFlow: Enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review* 38, 2 (2008), 69–74.
- [82] C. Mechalikh, H. Taktak, and F. Moussa. 2020. A fuzzy decision tree based tasks orchestration algorithm for edge computing environments. *Advances in Intelligent Systems and Computing* 1151 AISC (2020), 193–203. https://doi.org/10.1007/978-3-030-44041-1_18
- [83] Peter Mell, Tim Grance, et al. 2011. The NIST definition of cloud computing. (2011).
- [84] Giovanni Merlini, Rustem Dautov, Salvatore Distefano, and Dario Brunero. 2019. Enabling workload engineering in edge, fog, and cloud computing through openstack-based middleware. *ACM Transactions on Internet Technology* 19, 2 (2019). <https://doi.org/10.1145/3309705>
- [85] Merriam-Webster. Orchestration | Definition of Orchestration. 2021. Retrieved Feb 17, 2021 from <https://www.merriam-webster.com/dictionary/orchestration>.
- [86] Rashid Mijumbi, Joan Serrat, Juan-Luis Gorricho, Niels Bouten, Filip De Turck, and Raouf Boutaba. 2015. Network function virtualization: State-of-the-art and research challenges. *IEEE Communications Surveys & Tutorials* 18, 1 (2015), 236–262.
- [87] Carla Mouradian, Fereshteh Ebrahimnezhad, Yassine Jebbar, Jasmeen Kaur Ahluwalia, Seyedeh Negar Afrasiabi, Roch H Glitho, and Ashok Moghe. 2020. An IoT platform-as-a-service for NFV-based hybrid cloud/fog systems. *IEEE Internet of Things Journal* 7, 7 (2020), 6102–6115.
- [88] Carla Mouradian, Diala Naboulsi, Sami Yangui, Roch H. Glitho, Monique J. Morrow, and Paul A. Polakos. 2018. A comprehensive survey on fog computing: State-of-the-art and research challenges. *IEEE Communications Surveys and Tutorials* 20, 1 (2018), 416–464.
- [89] MQTT. Message Queuing Telemetry Transport. 2021. Retrieved Feb 17, 2021 from <https://mqtt.org/>.
- [90] Mithun Mukherjee, Rakesh Matam, Lei Shu, Leandros Maglaras, Mohamed Amine Ferrag, Nikumani Choudhury, and Vikas Kumar. 2017. Security and privacy in fog computing: Challenges. *IEEE Access* 5 (2017), 19293–19304.
- [91] Mithun Mukherjee, Lei Shu, and Di Wang. 2018. Survey of fog computing: Fundamental, network applications, and research challenges. *IEEE Communications Surveys and Tutorials* 20, 3 (2018), 1826–1857.
- [92] Mohammed Islam Naas, Philippe Raipin Parvedy, Jalil Boukhobza, and Laurent Lemarchand. 2017. IFogStor: An IoT data placement strategy for fog infrastructure. *Proceedings - 2017 IEEE 1st International Conference on Fog and Edge Computing, ICFEC 2017 May 2018* (2017), 97–104.
- [93] Ranesh Kumar Naha, Saurabh Garg, Dimitrios Georgakopoulos, Prem Prakash Jayaraman, Longxiang Gao, Yong Xiang, and Rajiv Ranjan. 2018. Fog computing: Survey of trends, architectures, requirements, and research directions. *IEEE Access* 6 (2018), 47980–48009.
- [94] Athanasios Naskos, Emmanouela Stachtiari, Anastasios Gounaris, Panagiotis Katsaros, Dimitrios Tsoumakos, Ioannis Konstantinou, and Spyros Sioutas. 2015. Dependable horizontal scaling based on probabilistic model checking. In *2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*. IEEE, 31–40.
- [95] S. B. Nath, S. Chattopadhyay, R. Karmakar, S. K. Addya, S. Chakraborty, and S. K. Ghosh. 2019. PTC: Pick-test-choose to place containerized micro-services in IoT. In *2019 IEEE Global Communications Conference (GLOBECOM)*. 1–6. <https://doi.org/10.1109/GLOBECOM38437.2019.9013163>
- [96] NGSI. NGSI 9/10 Information Model. 2012. Retrieved Feb 17, 2021 from http://www.openmobilealliance.org/release/NGSI/V1_0-20120529-A/OMA-TS-NGSI_Context_Management-V1_0-20120529-A.pdf.

- [97] Phu Nguyen, Nicolas Ferry, Gencer Erdogan, Hui Song, Stéphane Lavirotte, Jean-Yves Tigli, and Arnor Solberg. 2019. Advances in deployment and orchestration approaches for IoT-a systematic review. In *2019 IEEE International Congress on Internet of Things (ICIoT)*. IEEE, 53–60.
- [98] Phu Hong Nguyen, Nicolas Ferry, Gencer Erdogan, Hui Song, Stéphane Lavirotte, Jean-Yves Tigli, and Arnor Solberg. 2019. A systematic mapping study of deployment and orchestration approaches for IoT. In *IoTBDS*. 69–82.
- [99] R. Nicholson, T. Ward, D. Baum, X. Tao, D. Conzon, and E. Ferrera. 2019. Dynamic fog computing platform for event-driven deployment and orchestration of distributed Internet of Things applications. In *2019 Third World Conference on Smart Trends in Systems Security and Sustainability (WorldS4)*. 239–246. <https://doi.org/10.1109/WorldS4.2019.8903975>
- [100] NOMAD. Nomad by HashiCorp. 2021. Retrieved Jul 31, 2021 from <https://www.nomadproject.io>.
- [101] OpenFog. 2017. OpenFog Consortium Architecture Working Group. (February 2017), 162 pages.
- [102] Per-Olov Östberg, James Byrne, Paolo Casari, Philip Eardley, Antonio Fernandez Anta, Johan Forsman, John Kennedy, Thang Le Duc, Manuel Noya Marino, Radhika Loomba, et al. 2017. Reliable capacity provisioning for distributed cloud/edge/fog computing applications. In *2017 European Conference on Networks and Communications (EuCNC)*. IEEE, 1–6.
- [103] C. Pahl, N. E. Ioini, S. Helmer, and B. Lee. 2018. An architecture pattern for trusted orchestration in IoT edge clouds. In *2018 Third International Conference on Fog and Mobile Edge Computing (FMEC)*. 63–70. <https://doi.org/10.1109/FMEC.2018.8364046>
- [104] G. Papathanail, I. Fotoglou, C. Demertzis, A. Pentelas, K. Sgouromitis, P. Papadimitriou, D. Spatharakis, I. Dimolitsas, D. Dechouiotis, and S. Papavassiliou. 2020. COSMOS: An orchestration framework for smart computation offloading in edge clouds. In *NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium*. 1–6. <https://doi.org/10.1109/NOMS47738.2020.9110294>
- [105] Mike P. Papazoglou and Dimitrios Georgakopoulos. 2003. Introduction: Service-oriented computing. *Commun. ACM* 46, 10 (2003), 24–28.
- [106] C. Peitz. 2003. Web services orchestration and composition. *Computer* 36, 10 (2003), 46–52.
- [107] Charith Perera, Yongrui Qin, Julio C. Estrella, Stephan Reiff-Marganiec, and Athanasios V. Vasilakos. 2017. Fog computing for sustainable smart cities: A survey. *ACM Computing Surveys (CSUR)* 50, 3 (2017), 1–43.
- [108] Kai Petersen, Robert Feldt, Shahid Mujtaba, and Michael Mattsson. 2008. Systematic mapping studies in software engineering. In *Ease*, Vol. 8. 68–77.
- [109] Antonio Pintus, Davide Carboni, Andrea Piras, and Alessandro Giordano. 2010. Connecting smart things through web services orchestrations. In *International Conference on Web Engineering*. Springer, 431–441.
- [110] Paul Pop, Bahram Zarrin, Mohammadreza Barzegaran, Stefan Schulte, Sasikumar Punnekkat, Jan Ruh, and Wilfried Steiner. 2021. The FORA fog computing platform for industrial IoT. *Information Systems* (2021), 101727.
- [111] Subhab Pradhan, Abhishek Dubey, Shweta Khare, Saideep Nannapaneni, Aniruddha Gokhale, Sankaran Mahadevan, Douglas C. Schmidt, and Martin Lehofner. 2018. CHARIOT: Goal-driven orchestration middleware for resilient IoT systems. *ACM Transactions on Cyber-Physical Systems* 2, 3, SI (2018). <https://doi.org/10.1145/3134844>
- [112] Carlo Puliafito, Enzo Mingozi, Francesco Longo, Antonio Puliafito, and Omer Rana. 2019. Fog computing for the Internet of Things: A survey. *ACM Transactions on Internet Technology (TOIT)* 19, 2 (2019), 1–41.
- [113] Ju Ren, Deyu Zhang, Shiwen He, Yaoxue Zhang, and Tao Li. 2019. A survey on end-edge-cloud orchestrated network computing paradigms: Transparent computing, mobile edge computing, fog computing, and cloudlet. *ACM Computing Surveys (CSUR)* 52, 6 (2019), 1–36.
- [114] Wayne Robbins. 1997. Implementation and performance issues in an object-oriented orchestration architecture. In *Proceedings of IEEE International Conference on Multimedia Computing and Systems*. IEEE, 628–629.
- [115] Rodrigo Roman, Javier Lopez, and Masahiro Mambo. 2018. Mobile edge computing, fog et al.: A survey and analysis of security threats and challenges. *Future Generation Computer Systems* 78 (2018), 680–698.
- [116] Michel J. F. Rosa, Aletéia P. F. Araújo, and Felipe L. S. Mendes. 2018. Cost and time prediction for efficient execution of bioinformatics workflows in federated cloud. In *2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*. IEEE, 1703–1710.
- [117] Farah Ait Salaht, Frédéric Desprez, and Adrien Lebre. 2020. An overview of service placement problem in fog and edge computing. *ACM Computing Surveys (CSUR)* 53, 3 (2020), 1–35.
- [118] A. Salem and T. Nadeem. 2017. LAMEN: Towards orchestrating the growing intelligence on the edge. In *2016 IEEE 3rd World Forum on Internet of Things, WF-IoT 2016*. 508–513. <https://doi.org/10.1109/WF-IoT.2016.7845453>
- [119] Daniele Santoro, Daniel Zozin, Daniele Pizzolli, Francesco De Pellegrini, and Silvio Cretti. 2017. Foggy: A platform for workload orchestration in a fog computing environment. In *2017 9th IEEE International Conference on Cloud Computing Technology and Science (CLOUDCOM)*. IEEE, 345 E. 47th St. New York, NY, 231–234. <https://doi.org/10.1109/CloudCom.2017.62>
- [120] Jose Santos, Tim Wauters, Bruno Volckaert, and Filip De Turck. 2018. Fog computing: Enabling the management and orchestration of smart city applications in 5G Networks. *ENTROPY* 20, 1 (2018). <https://doi.org/10.3390/e20010004>

- [121] Subhadeep Sarkar and Sudip Misra. 2016. Theoretical modelling of fog computing: A green computing paradigm to support IoT applications. *Iet Networks* 5, 2 (2016), 23–29.
- [122] Mahadev Satyanarayanan, Victor Bahl, Ramón Caceres, and Nigel Davies. 2009. The case for vm-based cloudlets in mobile computing. *IEEE Pervasive Computing* (2009).
- [123] Mahadev Satyanarayanan, Zhuo Chen, Kiryong Ha, Wenlu Hu, Wolfgang Richter, and Padmanabhan Pillai. 2014. Cloudlets: At the leading edge of mobile-cloud convergence. In *6th International Conference on Mobile Computing, Applications and Services*. IEEE, 1–9.
- [124] Sukhpal Singh and Inderveer Chana. 2016. Cloud resource provisioning: Survey, status and future research directions. *Knowledge and Information Systems* 49, 3 (2016), 1005–1069.
- [125] Olena Skarlat, Stefan Schulte, Michael Borkowski, and Philipp Leitner. 2016. Resource provisioning for IoT services in the fog. In *2016 IEEE 9th International Conference on Service-Oriented Computing and Applications (SOCA)*. IEEE, 32–39.
- [126] OpenFlow Switch Specification. 2013. Open networking foundation. *ONF TS-015, Version 1, 3* (2013), 1–164.
- [127] Ivan Stojmenovic and Sheng Wen. 2014. The fog computing paradigm: Scenarios and security issues. In *2014 Federated Conference on Computer Science and Information Systems*. IEEE, 1–8.
- [128] Salman Taherizadeh and Vlado Stankovski. 2017. Auto-scaling applications in edge computing: Taxonomy and challenges. In *Proceedings of the International Conference on Big Data and Internet of Thing*. 158–163.
- [129] Antero Taivalsaari and Tommi Mikkonen. 2017. A roadmap to the programmable world: Software challenges in the IoT era. *IEEE Software* 34, 1 (2017), 72–80.
- [130] Bo Tang, Zhen Chen, Gerald Hefferman, Tao Wei, Haibo He, and Qing Yang. 2015. A hierarchical distributed fog computing architecture for big data analysis in smart cities. *Proceedings of the ASE BigData & Social Informatics 2015 October* (2015), 28.
- [131] Klervie Tocze and Simin Nadjm-Tehrani. 2018. A taxonomy for management and optimization of multiple resources in edge computing. *Wireless Communications and Mobile Computing* 2018 (2018).
- [132] Klervie Tocze and Simin Nadjm-Tehrani. 2019. ORCH: Distributed orchestration framework using mobile edge devices. In *2019 IEEE 3rd International Conference on Fog and Edge Computing, ICFEC - Proceedings*. 345 E 47th St. New York, NY, 1–10. <https://doi.org/10.1109/CFEC.2019.8733152>
- [133] Adel Nadjaran Toosi, Redowan Mahmud, Qinghua Chi, and Rajkumar Buyya. 2019. Management and orchestration of network slices in 5G, fog, edge and clouds. *Fog and Edge Computing* 10 (2019).
- [134] J. Tsai, I. Chuang, J. Liu, Y. Kuo, and W. Liao. 2020. QoS-aware fog service orchestration for industrial Internet of Things. *IEEE Transactions on Services Computing* (2020), 1. <https://doi.org/10.1109/TSC.2020.2978472>
- [135] TW van den Berg and A . Cramp. 2018. Container orchestration environments for M&S. In *2018 Winter Simulation Innovation Workshop, SIW 2018, 2018 Winter Simulation Innovation Workshop, SIW 2018, 21 January 2018 through 26 January 2018*. SISO-Simulation Interoperability Standards Organization.
- [136] Frank Van Lingen, Marcelo Yannuzzi, Anuj Jain, Rik Irons-Mclean, Oriol Lluch, David Carrera, J. L. Perez, Alberto Gutierrez, Diego Montero, Josep Marti, Ricard Maso, and J. P. Rodriguez. 2017. The unavoidable convergence of NFV, 5G, and Fog: A model-driven approach to bridge cloud and edge. *IEEE Communications Magazine* 55, 8 (2017), 28–35. <https://doi.org/10.1109/MCOM.2017.1600907>
- [137] Luis M. Vaquero, Felix Cuadrado, Yehia Elkhatib, Jorge Bernal-Bernabe, Satish N Srivama, and Mohamed Faten Zhani. 2019. Research challenges in nextgen service orchestration. *Future Generation Computer Systems* 90 (2019), 20–38.
- [138] Luis M. Vaquero and Luis Rodero-Merino. 2014. Finding your way in the fog: Towards a comprehensive definition of fog computing. *SIGCOMM Comput. Commun. Rev.* 44, 5 (Oct. 2014), 27–32. <https://doi.org/10.1145/2677046.2677052>
- [139] Luis M. Vaquero, Luis Rodero-Merino, Juan Caceres, and Maik Lindner. 2008. A break in the clouds: Towards a cloud definition. *ACM SIGCOMM Computer Communication Review* 39, 1 (2008), 50–55.
- [140] Karima Velasquez, David Perez Abreu, Marcio RM Assis, Carlos Senna, Diego F. Aranha, Luiz F. Bittencourt, Nuno Laranjeiro, Marilia Curado, Marco Vieira, Edmundo Monteiro, et al. 2018. Fog orchestration for the Internet of Everything: State-of-the-art and research challenges. *Journal of Internet Services and Applications* 9, 1 (2018), 14.
- [141] Karima Velasquez, David Perez Abreu, Marilia Curado, and Edmundo Monteiro. 2017. Service placement for latency reduction in the Internet of Things. *Annales des Telecommunications/Annals of Telecommunications* 72, 1-2 (2017), 105–115. <https://doi.org/10.1007/s12243-016-0524-9>
- [142] K. Velasquez, D. P. Abreu, D. Goncalves, L. Bittencourt, M. Curado, E. Monteiro, and E. Madeira. 2017. Service orchestration in fog environments. In *Proceedings - 2017 IEEE 5th International Conference on Future Internet of Things and Cloud, FiCloud 2017*, Vol. 2017-Janua. 329–336. <https://doi.org/10.1109/FiCloud.2017.49>
- [143] Alvaro Videla and Jason J.W. Williams. 2012. *RabbitMQ in Action: Distributed Messaging for Everyone*. Manning.
- [144] Alexandre Viejo and David Sánchez. 2019. Secure and privacy-preserving orchestration and delivery of fog-enabled IoT services. *Ad Hoc Networks* 82 (2019), 113–125.

- [145] Hariharasudhan Viswanathan, Parul Pandey, and Dario Pompili. 2016. Maestro: Orchestrating concurrent application workflows in mobile device clouds. IEEE, 257–262.
- [146] Sebastian Wahle, Thomas Magedanz, and Frank Schulze. 2012. The OpenMTC framework- M2M solutions for smart cities and the Internet of Things. In *2012 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*. IEEE, 1–3.
- [147] Florian Wamser, Chiara Lombardo, Constantinos Vassilakis, Lam Dinh-Xuan, Paolo Lago, Roberto Bruschi, and Phuoc Tran-Gia. 2018. Orchestration and monitoring in fog computing for personal edge cloud service support. In *2018 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN)*. 345 E. 47th St. New York, NY, 91–96. <https://doi.org/10.1109/LANMAN.2018.8475113>
- [148] Nan Wang, Blesson Varghese, Michail Matthaiou, and Dimitrios S Nikolopoulos. 2017. ENORM: A framework for edge node resource management. *IEEE Transactions on Services Computing* (2017).
- [149] Denis Weerasiri, Moshe Chai Barukh, Boualem Benatallah, Quan Z. Sheng, and Rajiv Ranjan. 2017. A taxonomy and survey of cloud resource orchestration techniques. *ACM Computing Surveys (CSUR)* 50, 2 (2017), 1–41.
- [150] Z. Wen, R. Yang, P. Garraghan, T. Lin, J. Xu, and M. Rovatsos. 2017. Fog orchestration for Internet of Things services. *IEEE Internet Computing* 21, 2 (2017), 16–24. <https://doi.org/10.1109/MIC.2017.36>
- [151] Zhenyu Wen, Renyu Yang, Peter Garraghan, Tao Lin, Jie Xu, and Michael Rovatsos. 2017. Fog orchestration for Internet of Things services. *IEEE Internet Computing* 21, 2 (2017), 16–24.
- [152] Cecil Wöbker, Andreas Seitz, Harald Mueller, and Bernd Bruegge. 2018. Fogernetes: Deployment and management of fog computing applications. IEEE, 1–7.
- [153] Zhang Yaoxue. 2004. Transparency computing: Concept, architecture and example. *Acta Electronica Sinica* 32, 12 A (2004), 169–174.
- [154] Shanhe Yi, Cheng Li, and Qun Li. 2015. A survey of fog computing: Concepts, applications and issues. In *Proceedings of the 2015 Workshop on Mobile Big Data*. 37–42.
- [155] E. Yigitoglu, L. Liu, M. Looper, and C. Pu. 2017. Distributed orchestration in large-scale IoT systems. In *Proceedings - 2017 IEEE 2nd International Congress on Internet of Things, ICIOT 2017*. 58–65. <https://doi.org/10.1109/IEEE.ICIOT.2017.16>
- [156] Emre Yigitoglu, Mohamed Mohamed, Ling Liu, and Heiko Ludwig. 2017. Foggy: A framework for continuous automated IoT application deployment in fog computing. 345 E. 47th St. New York, NY, 38–45. <https://doi.org/10.1109/AIMS.2017.14>
- [157] Ashkan Yousefpour, Caleb Fung, Tam Nguyen, Krishna Kadiyala, Fatemeh Jalali, Amirreza Niakanlahiji, Jian Kong, and Jason P. Jue. 2019. All one needs to know about fog computing and related edge computing paradigms: A complete survey. *Journal of Systems Architecture* December 2018 (2019).
- [158] L. Zanzi, F. Giust, and V. Sciancalepore. 2018. M2EC: A multi-tenant resource orchestration in multi-access edge computing systems. In *2018 IEEE Wireless Communications and Networking Conference (WCNC)*, Vol. 2018-April. 1–6. <https://doi.org/10.1109/WCNC.2018.8377292>
- [159] Aleksandr Zavodovski, Nitinder Mohan, Suzan Bayhan, Walter Wong, and Jussi Kangasharju. 2019. ExEC: Elastic extensible edge cloud. In *Proceedings of the 2Nd International Workshop on Edge Systems, Analytics and Networking*. 24–29.

Received March 2021; revised August 2021; accepted September 2021