Application Management in Fog Computing Environments: A Taxonomy, Review and Future Directions

REDOWAN MAHMUD, KOTAGIRI RAMAMOHANARAO, and RAJKUMAR BUYYA,
University of Melbourne

The Internet of Things (IoT) paradigm is being rapidly adopted for the creation of smart environments in various domains. The IoT-enabled cyber-physical systems associated with smart city, healthcare, Industry 4.0 and Agtech handle a huge volume of data and require data processing services from different types of applications in real time. The Cloud-centric execution of IoT applications barely meets such requirements as the Cloud datacentres reside at a multi-hop distance from the IoT devices. *Fog computing*, an extension of Cloud at the edge network, can execute these applications closer to data sources. Thus, Fog computing can improve application service delivery time and resist network congestion. However, the Fog nodes are highly distributed and heterogeneous, and most of them are constrained in resources and spatial sharing. Therefore, efficient management of applications is necessary to fully exploit the capabilities of Fog nodes. In this work, we investigate the existing application management strategies in Fog computing and review them in terms of architecture, placement and maintenance. Additionally, we propose a comprehensive taxonomy and highlight the research gaps in Fog-based application management. We also discuss a perspective model and provide future research directions for further improvement of application management in Fog computing.

CCS Concepts: • **General and reference** → **General literature**; • **Computer systems organization** → **Distributed architectures**;

Additional Key Words and Phrases: Fog computing, Internet of Things, application architecture, application placement, application maintenance

ACM Reference format:
Redowan Mahmud, Kotagiri Ramamohanarao, and Rajkumar Buyya. 2020. Application Management in Fog Computing Environments: A Taxonomy, Review and Future Directions. *ACM Comput. Surv.* 53, 4, Article 88 (July 2020), 43 pages.
https://doi.org/10.1145/3403955

1 INTRODUCTION

The Internet of Things (IoT) concept has changed the structure of material environments by connecting numerous computing components, digital machines, dumb objects and animals with the Internet. IoT enables them to perceive the external ambiance as sensors and trigger any action based on the given commands using actuators [Gubbi et al. 2013]. Thus, IoT creates a novel type of

Authors' address: R. Mahmud, K. Ramamohanarao, and R. Buyya, Cloud Computing and Distributed Systems (CLOUDS) Laboratory, School of Computing and Information Systems, Floor: 07, Room: 7.22, Doug Mcdonell Building, Grattan St, Parkville Campus, The University of Melbourne, Melbourne, Victoria, 3052, Australia; emails: mahmudm@student.unimelb.edu.au, rkotagiri@gmail.com, rbuyya@unimelb.edu.au.

0360-0300/2020/07-ART88 $15.00
https://doi.org/10.1145/3403955

interaction among different real-world entities in ingenious ways. The ongoing advancement in the field of hardware and communication technology is consistently improving and expanding the applicability of IoT that consequently helps in realizing the theory of smart city, remote healthcare, Industry 4.0 and Agtech [Mahmud et al. 2018a]. Recently, various cyber-physical systems (CPSs) for smart environments such as indoor locator, digital health recorder and robot-assisted supply chain manager have been developed through the widespread deployment of IoT devices. Moreover, according to the current trend of practising IoT, it is expected that by 2030, there will be 1.2 trillion active IoT devices with potential economic impact of $15 trillion per year [Chowdhury et al. 2019].

IoT devices can generate data incessantly or periodically. These data are filtered, analysed and evaluated by different types of applications [Belli et al. 2015]. As most of the IoT devices are equipped with limited energy, computing and networking capabilities, they are considered ill suited to execute heavyweight applications [Martinez et al. 2015]. Moreover, based on the working environments of IoT-enabled CPSs, corresponding applications often require data processing within a defined time frame. Their data-driven interactions with the IoT devices also demands a less-congested network. The computing infrastructure executing the applications for IoT-enabled CPSs needs to observe these issues so that the desired responsiveness of the CPSs can be ensured [Afrin et al. 2018].

1.1 Scope and Benefits of Fog Computing

Cloud computing has been the backbone for hosting and offering subscription-oriented computing and application services. It is also used to execute the applications for different IoT-enabled CPSs [Yannuzzi et al. 2014]. The Cloud datacentres consist of data and computing servers to facilitate users with storage and virtualized computing instances [Buyya et al. 2009]. Nevertheless, these datacentres are located at a multi-hop distance from the IoT devices. Therefore, a longer period of time is required to transfer data and command between the IoT devices and the applications executing on the Cloud instances that also degrades the application service delivery time [Afrin et al. 2015]. Furthermore, when a large number of IoT devices initiate data-driven interactions with remote applications, a substantial load is added to the network and severe congestion occurs. The computational overhead on Cloud datacentres also increases [Madsen et al. 2013]. Because of these limitations, the Cloud-centric application execution model often fails to meet the service requirements of different IoT-enabled CPSs. To address them, an extension of Cloud computing named *Fog computing* was introduced by CISCO in 2012 [Bonomi et al. 2012].

Fog manages an intermediate layer between end user devices and Cloud datacentres by utilizing the computing components within the edge network [Yousefpour et al. 2019]. In Fog environments, these computing components, such as personal computers, gateways, Raspberry PIs, nano-servers and micro-datacentres, are commonly known as *Fog nodes*. As shown in Figure 1, the Fog nodes execute various IoT applications in proximity of data sources. Hence, Fog computing resists a huge amount of data from sending towards Cloud datacentres and decreases the data propagation delay. Consequently, the service latency of different applications improves [Yao and Ansari 2019]. Moreover, Fog computing conserves network bandwidth that reduces the scope of network congestion. Through Fog computing, providers migrate the computational load from Cloud datacentres to network edge. As Fog nodes are less expensive, Fog computing lowers the operational cost of providers, saves energy for the Cloud datacentres and improves the quality of experience (QoE) of users. Additionally, Fog supports robust location awareness to simplify the communication with mobile and energy constrained end user devices [Puliafito et al. 2019].

On the basis of the aforementioned features, Fog computing is considered very promising compared to Cloud in meeting the application service requirements of different IoT-enabled CPSs.

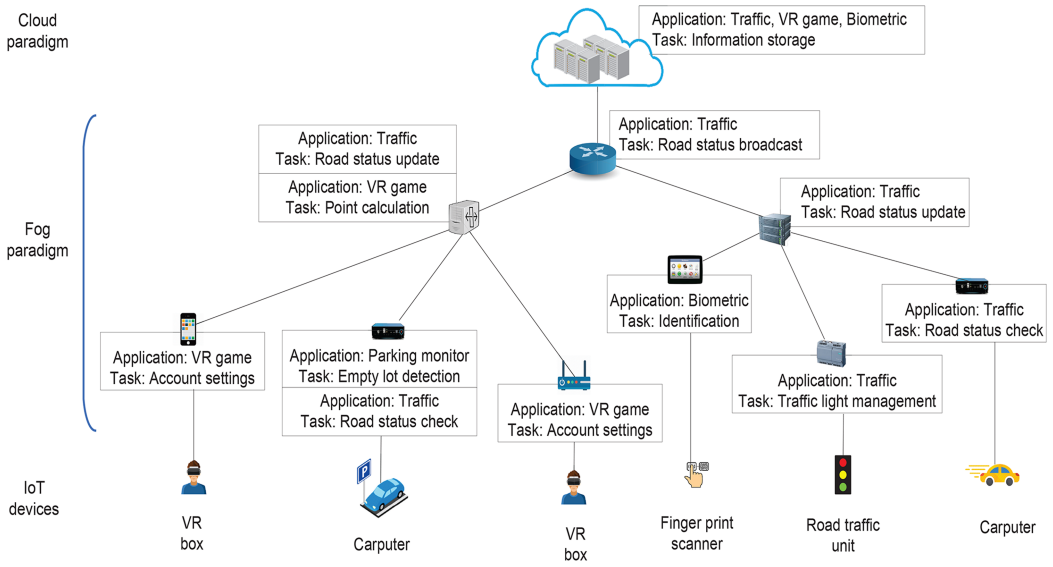

Fig. 1. An illustration of application execution in Fog environments.

Therefore, several technology giants, such as Amazon, Alphabet and Microsoft, have already started integrating Fog computing with their Cloud infrastructure [IoT for All 2018]. In addition, there are other companies, such as Sonm, NEC Laboratories, FogHorn Systems and Drofika Labs, that intend to make software systems for Fog environments [Syed et al. 2016]. The development of the FogFlow framework is regarded as a successful attempt in this direction [Cheng et al. 2018]. There also exists a Fog-based real-time patient monitoring system developed by Tata Consultancy Services [Maksimović 2018].

1.2 Application Management in Fog Computing

Fog computing creates a wide distribution of infrastructure and platform services. Infrastructure services include on-demand exploitation of computing, networking (bandwidth and firewalls) and storage resources, whereas platform services facilitate application runtime environments, operating systems and programming interfaces [Dastjerdi et al. 2016]. Fog resource management denotes administrative operations such as deployment, virtualization and monitoring of Fog nodes that foster the Fog-based infrastructure and platform services [Sarkar et al. 2015]. Additionally, Fog resource management realizes load balancing, dynamic provisioning and auto-scaling to ensure service availability and multi-tenancy [Hong and Varghese 2018].

Efficient Fog resource management assists IoT-enabled CPSs to operate multiple applications simultaneously. However, the characteristics of these applications vary from one CPS to another. For example, the expected application service delivery time for a CPS that remotely monitors the respiratory functions of critical patients is quite stringent compared to a CPS which measures the environmental parameters [Tuli et al. 2019]. Moreover, an application that assists a CPS to perform virtual reality operations handles a huge amount of data in per unit time compared to an application which helps in tracking the empty parking slots. Such diversified characteristics play vital roles in defining the quality of service (QoS) requirements of the applications that cannot be met only through Fog resource management [Mahmud et al. 2019a]. This perception also urges to develop different application management strategies according to the preferences of the applications. Usually, an application management strategy refers to a collection of algorithms, mathematical

models, empirical analysis and recommendations that regulate the implementation, installation and execution of applications in a computing environment. Moreover, application management strategies practice admission control, location transparency, data maintenance and service resiliency as per the demands of the corresponding system [Varshney and Simmhan 2017]. Nevertheless, three research questions become crucial while developing application management strategies for Fog computing environments:

- *How should the applications be composed?*
  To address this question, an application management strategy requires to specify the features of applications, such as their programming model, functional layout, service type and workload type so that they can be aligned with the Fog-based infrastructure and platform services.
- *How should the applications be placed?*
  To address this question, an application management strategy requires to find suitable placement options for the applications in Fog environments. At the same time, the strategy needs to make a balance between application-centric QoS requirements.
- *How should the applications be maintained?*
  To address this question, an application management strategy requires to facilitate security and resiliency support during application execution in Fog environments. Moreover, the strategy needs to monitor the performance of the applications in a consistent manner.

Although operational responses to these questions are important for efficient application management, their actual realization in Fog computing is a very challenging task.

1.2.1 Challenges of Application Management. The challenges faced by application management strategies in Fog environments are listed as follows:

- *Resource and energy constrained, distributed and heterogeneous Fog nodes*: Most of the Fog nodes are constrained in processing power, networking capability, storage and energy capacity. They are deployed in distributed order at the edge network. Their communication standards and operating systems also vary from one to another [Madsen et al. 2013]. As a consequence, the time-optimized and platform independent application management become tedious to ensure in Fog. Additionally, Fog infrastructure is less flexible than Cloud in terms of sharing resources—for example, the Fog nodes located in California cannot be harnessed for the CPSs in Melbourne. Such constraints limit the execution domain for large-scale IoT applications in Fog [Lee et al. 2019].
- *Subjected to uncertain failures*: Fog nodes are highly prone to get affected by anomalies such as power failures and out of capacity faults that obstruct the execution of applications assigned to them [Melnik et al. 2019]. Due to latency issues, the recovery of applications also becomes difficult.
- *Standard-less integration*: The applications executing in Fog often need the services offered by different computing paradigms. For example, the Fog-based health data analytic applications require the Cloud-based storage service to facilitate location-independent medical report sharing. In such scenarios, integration of Fog infrastructure with others is necessary [Deng et al. 2016]. Nevertheless, the absence of efficient frameworks and standards resist the Fog environments to provide this assistance to the applications.
- *Lack of interoperability*: The structural differences between Fog and Cloud environments obstruct the interoperability of IoT applications. Due to lack of interoperability, an extensive programming effort is required to customize the existing Cloud-based IoT applications so that they can leverage the benefits of Fog computing [Mahmud et al. 2018a].

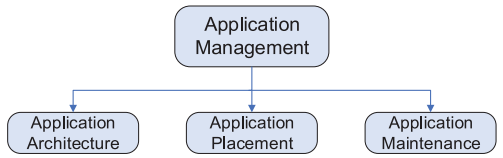

Fig. 2. Different aspects of application management.

- *Absence of business model*: Fog environments lack business models that hampers the budget management of users and the profit enhancement of providers. These monetary issues consequently resist both parties to execute applications in Fog [Kim and Chung 2018].
- *Inefficient task distribution*: Fog environments operate in a decentralized manner across the edge network. The coexistence of multiple decision-making entities increases the application management complexity in Fog environments that ultimately results in poor distribution of application tasks over the Fog nodes [Baccarelli et al. 2017].
- *Less secured*: The outcomes of applications executing in Fog can be requested by different types of users. For example, the results of a Fog-based healthcare application are relevant to the hospitals, insurance companies and employer organizations. In such cases, despite the necessity, the on-demand and secured access to application outcomes becomes difficult to ensure because of the resource scarcity and orientation of Fog environments [Shirazi et al. 2017].

1.2.2 Motivation of Research. Considering the associated challenges, a notable number of application management strategies have already been developed for Fog computing environments. They predominantly focus on the modularization of applications to deal with the resource constraints of Fog nodes [Benamer et al. 2018; Fiandrino et al. 2019; Prabavathy et al. 2019]. These strategies also adopt the web service–based communication techniques to simplify the interactions between different components of modular applications hosted on distributed Fog nodes [Mass et al. 2018; Wiener et al. 2019]. While assigning the applications to the Fog nodes, the existing application management strategies give much emphasis on meeting the service delivery deadline and optimizing the cost and energy consumptions [Luo et al. 2019; Gazori et al. 2019; Huang et al. 2019]. Most of them operate discretely and apply strict synchronization measures over the Fog nodes to mitigate the effect of interference [Mahmud et al. 2020; Wang et al. 2019]. The application management strategies also incorporate both proactive and reactive fault tolerance techniques to support the reliable execution of the applications in Fog environments [Anglano et al. 2019b; Filiposka et al. 2019; Noura et al. 2019]. However, in the literature, very few initiatives have been found that categorize the application management strategies in a systematic way [Yousefpour et al. 2019; Naha et al. 2018]. Therefore, in this work, we identify three important aspects of application management in Fog computing environments, namely application architecture, application placement and application maintenance, as shown in Figure 2, and present a separate taxonomy for each of them. Based on the proposed taxonomy, we also review the existing application management strategies and denote how the research community can leverage the solutions to make further progress. The main contributions of this work include the following:

- We review the existing literature on application management strategies in Fog from the perspectives of architecture, placement and maintenance and propose their taxonomy.
- We discuss a framework that is logically distributed and helps adaptive and holistic management of applications in Fog computing environments.

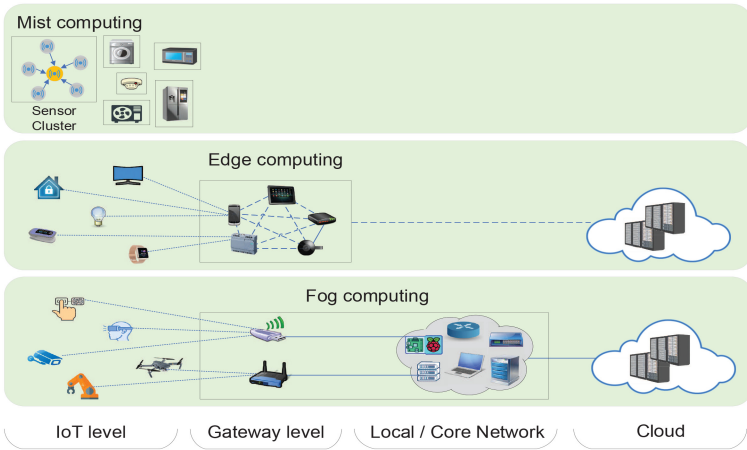

Fig. 3. Domain of Mist, Edge and Fog computing.

- We identify the research gaps in Fog computing–based application management and highlight future research directions for further improvement in this field.

1.3 Article Organization

The rest of the article is organized as follows. The differences between Fog and other contemporary computing paradigms along with the description of related surveys are illustrated in Section 2. In Section 3, a discussion on application architecture in Fog environments is presented. Section 4 highlights the existing techniques to place applications in Fog environments. Section 5 explores the application maintenance operations. Section 6 demonstrates a perspective framework for Fog-based application management. Section 7 provides future directions to improve the application management strategies in Fog environments based on the identified research gaps in Sections 4 and 5. Finally, Section 8 summarizes our efforts and concludes the survey.

2 BACKGROUND STUDY

2.1 Comparison among Mist, Edge and Fog computing

Fog, Edge and Mist computing support application execution in proximity of data sources as shown in Figure 3. More precisely, Mist computing enables the IoT devices to process data within themselves, whereas Edge computing performs the processing operations at the gateways of IoT devices [Shi et al. 2016; Preden et al. 2015]. For instance, smart watches can be considered as IoT devices. End users usually connect smart watches to their smart phones via Bluetooth Low Energy (LE) networking so that they can receive mobile notifications while walking or driving. Here, the smart phones act as the IoT gateways for the smart watches. At the same time, the smart watches sense blood pressure, heartbeat and oxygen saturation rate of the users. When a smart watch executes the applications to process its generated data, Mist computation occurs [Barik et al. 2018]. Conversely, Edge computation happens when the smart watch forwards the data to a smart phone–based application for processing [Shi and Dustdar 2016]. However, compared to them, Fog computing not only harnesses the IoT gateways but also engages other computing components within the edge network, such as smart routers, personal computers, Raspberry Pi devices and even micro-datacentres, to process the IoT data [Madsen et al. 2013].

Although Mist and Edge computing can solve many IoT-related issues, they have certain limitations. The computing components in Mist are not abundant in processing, networking and energy

Table 1. Summary of Mist, Edge and Fog Computing

Facts	Mist	Edge	Fog
Place of operation	IoT devices	Gateway devices	Specialized networking and computing machines
Elementary hardware	Microcontroller	Programmable logic controller	Single-board computer
Wireless standards	Zigbee and Bluetooth LE	Bluetooth and WiFi	WiFi and LTE
Policy manager	Manufacturer	Users	Service providers
Application deployment	Programmed	Installed by user	Requested by user to service providers
Resource assignment	Dedicated	Shared	Shared or virtualized
Application-user mapping	Single application, single user	Multiple application, single user	Multiple application, multiple user
Resource orientation	Stand-alone, homogeneous cluster	Peer to peer, ad hoc	Cloud of Things
Cloud communication	Incoherent or through mediator	Event driven	Seamless
Fault tolerance techniques	Replacement	User-defined exception handling	Proactive and reactive
Extended from	Wireless sensor network, embedded systems	Personalized computing environments	Cloud computing

capacity. They are less capable of executing large-scale and complex applications for a longer period of time [Uehara 2018]. However, the management of Edge nodes are very much user centric, incorporating only reactive fault-tolerant facilities. In Edge environments, fairness is also tedious to ensure among multiple users [Satyanarayanan 2017]. Fog computing overcomes these limitations of Mist and Edge by leveraging comparatively powerful resources at the user premises level and lessening the burdens of resource and application service management from the users. Moreover, Fog computing maintains a seamless communication with Cloud datacentres that eventually offers an extensive execution platform for the IoT applications [Mahmud et al. 2018b]. The notable differences between Mist, Edge and Fog computing are listed in Table 1.

However, Mist, Edge and Fog are relatively new computing paradigms, and their evolution processes are ongoing. Therefore, many researchers and industries adopt different approaches to define them. For instance, there are several research works in the literature that consider Edge computing as a subset of Fog computing [Tuli et al. 2019], whereas in other works, Edge computing is regarded as a superset embracing all paradigms where the computation is moved to the edge network, including Fog computing, Mobile Cloud computing (MCC) and Mobile Edge computing (MEC) [Shi et al. 2016]. There are also some examples where Fog and Edge computing are used interchangeably [Chiti et al. 2019]. Moreover, in certain cases, Edge computation is regarded as a service model which is offered by different paradigms, namely Dew, Mist and Fog computing. According to this concept, Dew computing happens in the IoT devices and Mist computing occurs at the IoT gateways [Cristescu et al. 2019]. Nevertheless, among these contemporary paradigms, Fog computing is considered highly feasible due to its widespread support for the IoT applications.

2.2 Related Surveys in Fog Computing

In the context of Fog computing, resource and application management are equally important. In fact, without efficient application service management, the capabilities of Fog resources cannot be

exploited completely and vice versa. Nevertheless, in existing Fog-based literature surveys, application management is considered as a part of Fog resource management. Among these surveys, Hu et al. [2017], Mouradian et al. [2017], Mahmud et al. [2018b] and Yousefpour et al. [2019] provide the general discussions on Fog computing. They review the research on Fog computing from an architectural perspective and highlight the key technologies and limitations of Fog computing. Moreover, they illustrate the benefits of Fog computing and clearly identify the differences between Fog and Cloud computing. Other Fog-based literature surveys, including those of Hong and Varghese [2018], Li et al. [2018a], Naha et al. [2018] and Ghobaei-Arani et al. [2019], explore the basic resource management approaches in Fog environments. They investigate various management frameworks, scheduling techniques and provisioning algorithms for Fog resources. Furthermore, they review the resource orchestration techniques in layered Fog environments and study the resource management policies in accordance with the application service requirements. Some other literature surveys exist that focus on a specific aspect of Fog resource management. For example, Osanaiye et al. [2017] address the virtual computing instances migration methods in Fog computing, and Baccarelli et al. [2017] inspect energy-efficient Fog resource management.

Moreover, Bellavista et al. [2019], Nath et al. [2018] and Puliafito et al. [2019] conduct surveys to conceptualize the application service management in Fog environments. They discuss the communication, security, data and actuation management as the parts of application management. In addition, they highlight different application-specific Fog architectures and give an overview to realize them for various IoT-enabled CPSs. Nevertheless, other literature surveys target particular Fog computing–based applications and their service management. For example, Aazam et al. [2018] study computation offloading techniques in Fog computing environments. Similary, Kraemer et al [2017], Mukherjee et al. [2018] and Perera et al. [2017] investigate the existing approaches that enable Fog computing in smart healthcare, advanced networking and smart city–based applications, respectively. However, the works of Roman et al. [2018], Shirazi et al. [2017] and Zhang et al. [2018] are among those literature surveys which discuss the security aspects of Fog computing from both resource and application management perspectives.

In Table 2, a summary of existing Fog literature surveys and their comparative study with respect to our work is presented. As noted, the existing surveys do not explore application management in Fog environments comprehensively. More specifically, they barely discuss application architecture, placement and maintenance in a collective manner and illustrate the literature taxonomy accordingly. In this work, we address these shortcomings. We also identify the associate research gaps, demonstrate a perspective framework for distributed application management and provide future directions for improvement of the Fog computing concept. The following sections of this work present a detailed review of existing application management strategies in Fog computing.

3 APPLICATION ARCHITECTURE

The complexities of executing IoT applications in distributed, heterogeneous and resource constrained Fog nodes can be addressed if the architecture of applications is defined as per the specifications of the corresponding Fog environment. An elastic architecture also helps interoperability between different versions of an application. Moreover, the elements of application architecture such as the programming model and workload type are used to determine the placement strategy and resource consumptions of the applications. The service type of an application denotes the scope of its external exposure that assists in application maintenance. Figure 4 provides a taxonomy on application architecture highlighting the main elements. These elements are described in the following.

Table 2. Summary of Literature Surveys in Fog Computing

Surveys	Issues					
	Discusses application architecture	Investigates application placement techniques	Explores application maintenance operations	Provides taxonomy on application management	Conceptualizes application management framework	Relates application and resource management
[Aazam et al. 2018]	∂	✓	∂		✓	
[Baccarelli et al. 2017]	✓			∂	✓	✓
[Bellavista et al. 2019]		✓		∂	✓	✓
[Ghobaei-Arani et al. 2019]		✓		∂		✓
[Hong and Varghese 2018]	∂	✓		∂		✓
[Hu et al. 2017]	∂	∂				✓
[Kraemer et al. 2017]	∂	✓				
[Li et al. 2018a]		∂	∂	∂		
[Mahmud et al. 2018b]	✓	✓	∂	∂		
[Mouradian et al. 2017]		∂			✓	✓
[Mukherjee et al. 2018]	∂	✓	✓		✓	
[Naha et al. 2018]		∂	∂	✓	✓	✓
[Nath et al. 2018]		∂	✓	∂	✓	
[Osanaiye et al. 2017]			∂		✓	✓
[Puliafito et al. 2019]	∂	✓				✓
[Perera et al. 2017]	✓	✓			✓	
[Roman et al. 2018]	∂		✓			✓
[Shirazi et al. 2017]			✓		✓	✓
[Yousefpour et al. 2019]		✓	✓	∂		
[Zhang et al. 2018]	∂		✓		✓	✓
This survey	✓	✓	✓	✓	✓	✓

Note: The ✓ denotes broad discussion on the respective issue.
Note: The ∂ denotes partial discussion on the respective issue.

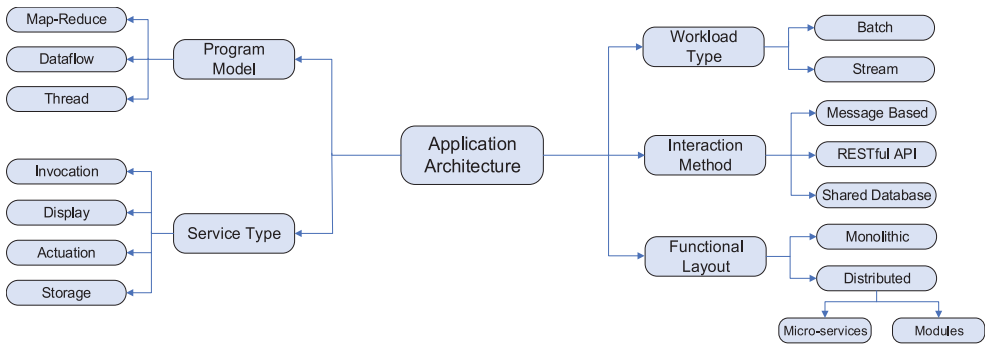

Fig. 4. Taxonomy on application architecture.

3.1 Functional Layout

An application performs different types of operations. For example, an image processing application reduces noises from an image, converts colors, extracts features and compares the results with predefined thresholds. The functional layout of an application defines the arrangement of these operations and assists in realizing the possible distribution of the application in constrained Fog environments. The functional layout of applications can be classified into two types: monolithic and distributed.

3.1.1 Monolithic. In monolithic applications, all computational operations are encapsulated in a single program. These applications function independently of each other. Within such applications, developer-specific parallelism techniques are applied so that they can run over multiple processing cores of the host Fog node. In the literature, Abbasi et al. [2019], Anglano et al. [2019b], Fizza et al. [2018], He et al. [2019] and Yousefpour et al. [2018] discuss the monolithic applications in Fog.

3.1.2 Distributed. The computational operations of a distributed application are organized in separate programs. Compared to monolithic applications, distributed applications are easier to expand. The programs of a distributed applications can be executed in a single Fog node or can be operated by several Fog nodes in a collaborative manner. Based on the dependency of computational operations, distributed applications are classified into two categories.

- *Module based*: In module-based distributed applications, the application programs are tightly coupled and dependent on each other. They are devotedly provisioned for serving data of a particular source. [AbdElhalim et al. [2019], Anzanpour et al. [2019], Ding et al. [2019], Djemai et al. [2019] and Prabavathy et al. [2019] discuss module-based distributed applications.
- *Micro-services*: Through micro-service-based implementation, the computational operations of an application are shared among different CPSs to process their data simultaneously. Unlike application modules, micro-services are loosely coupled and function independently. Due to explicit isolation, a micro-service of an application can be easily attached to other applications as per the requirements. Kayal and Liebeherr [2019], Lera et al. [2019], Mohamed et al. [2019], Wiener et al. [2019] and Zhu et al. [2019] highlight the micro-service-based implementation of IoT applications.

3.2 Program Model

The program model defines the execution order of computing operations in an application and guides to provision resources for application as per their dimension and predicted life cycle. Three different types of program models have been widely adopted while developing the applications in Fog.

3.2.1 Thread. To ensure simultaneous execution of independent computational operations within an application, the thread program model is used. It is one of the primitive program models that helps achieve concurrency in resource constrained Fog nodes. Anglano et al. [2019a], Chiti et al. [2019], Al-Khafajiy et al. [2019], Liu et al. [2019], Li et al. [2018b] and Tychalas and Karatza [2020] follow the thread model to develop applications for Fog computing environments. The advanced versions of the thread model such as Map-Reduce and dataflow are also used predominantly in Fog computing.

3.2.2 Map-Reduce. Through this model, the large-volume inputs for an application are divided into multiple chunks so that all operations can run in parallel over the given inputs. Later, the

processing outcomes of each chunk are combined to generate the overall output of the application. Such a program model for Fog-based applications is discussed in the work of Yue et al. [2018], Li et al. [2017], Jeong et al. [2017], Pang et al. [2018], Imai et al. [2018] and Dang et al. [2018].

3.2.3 Dataflow. In the dataflow program model, the output of a computational operation is fed to another operation as input, and this process continues for the subsequent operations. This program model binds all computational operations of an application through a directed acyclic graph (DAG). The size of input data handled through this model is not usually as large as that of the Map-Reduce model. In the literature, NAAS et al. [2018], Oma et al. [2019], Ozeer et al. [2018], Suter et al. [2019] and Guerrero et al. [2019] consider the dataflow program model for the applications.

3.3 Service Type

The service type of an application refers to its outcome that is delivered to the corresponding and requesting CPSs. The size of output of an application depends on its service types, which helps model the data propagation delay. Based on the physical environment, application service outcomes can vary. The services of different IoT applications can be classified into four types.

3.3.1 Invocation. An IoT application can invoke the execution of another application as its service. For example, the IoT application monitoring a forest fire can initiate a robotic application to meet the emergency requirements. Usually, in this type of service, an initiation command along with necessary arguments are forwarded from the source application to the requested application. Anzanpour et al. [2019], Fadahunsi and Maheswaran [2019] and Mohamed et al. [2019] discuss invocation as a service type of the applications.

3.3.2 Display. There exist several IoT applications, such as virtual reality gaming and smart surveillance, that visualize the service outcomes to the users. The quality of such application services explicitly depends on the networking condition between the end users and the associated Fog nodes. In the literature, Arkian et al. [2017], Benamer et al. [2018], Elbamby et al. [2018], Yin et al. [2018] and Zeng et al. [2018] highlight display-based services of applications in Fog environments.

3.3.3 Actuation. After processing incoming data, several IoT applications trigger actuators to initiate the required physical action. For example, the remote patient monitoring system can actuate the oxygen supply engine during emergency situations. Abbasi et al. [2019], Dehnavi et al. [2019], Kim et al. [2019] and Yue et al. [2018] consider actuation as a service type for the applications.

3.3.4 Storage. For long-term data collection or crowd sourcing, IoT applications are often used. These applications aggregate these data and store them in Cloud or Fog nodes for future analysis by other applications. AbdElhalim et al. [2019], Concone et al. [2019] and Karatas and Korpeoglu [2019] mention storage as an application service in Fog computing.

3.4 Interaction Method

While processing data in a collaborative manner, the computing operations and applications require interactions with each other to share and store the intermediate outcomes for further usage. However, such interactions become crucial when operating across multiple Fog nodes. In the following sections, different interaction methods for Fog-based applications are discussed.

3.4.1 Shared Database. The shared database is one of the primitive methods of sharing data. In this method, data are stored in a particular location, and all applications and computing operations requiring the data have direct access to it. This method also supports multi-level data distribution

from local and global perspectives for large-scale systems. The shared database for Fog-based applications is discussed in the work of Arkian et al. [2017], Karatas and Korpeoglu [2019], Nasir et al. [2019] and Suter et al. [2019].

3.4.2 Message Based. In this method, the host Fog nodes of computing operations or applications exchange lightweight messages to notify the current state of data processing. In most of the cases, this message transmission is supervised by a dedicated entity and follows the publish and subscribe protocol for machine to machine communication. Unlike shared database–driven interaction, this method is often used for small-scale systems. In the literature, Luo et al. [2019], Vinuenzo Naranjo et al. [2018], Skarlat et al. [2017] and Ozeer et al. [2018] discuss message-based interactions in Fog.

3.4.3 Representational State Transfer. Representational state transfer establishes a web service–based communication between the host Fog nodes using the http protocol. Representational state transfer allows data exchange through several stateless commands, such as get and post, and often follows the push and pull approach during device-level interactions. Mass et al. [2018], Santos et al. [2019], Zeng et al. [2018], Tuli et al. [2019] and Mahmud et al. [2018c] consider representational state transfer in their work. Due to the speed and ease of scalability, this method is being widely used by IoT applications compared to shared database and message-based interactions.

3.5 Workload Type

Workload denotes the characteristics of input processed by an application. The knowledge of workload is very important to select the host Fog node that has the appropriate configurations. There are two types of workload for IoT applications: batch and stream.

3.5.1 Batch. The bundle of non-interactive inputs for an application is often regarded as the batch workload. Once accumulated from multiple sources, the batch workload is submitted to the application for processing. The dispatch order of the inputs in such a workload can be shuffled as per the availability of resources to ensure the desired performance of the application. In the literature, Mouradian et al. [2019], Yue et al. [2018], Wang et al. [2019a], Wang et al. [2019b], Tychalas and Karatza [2020] and Zhu et al. [2019] discuss the batch workload for Fog-based applications.

3.5.2 Stream. This type of workload is generated by different sources in a periodic manner. Therefore, while developing real-time IoT solutions, the stream workload is preferred over the batch workload. The specification and processing requirements of the stream workload can change with the course of time based on the sensing frequency of associated IoT devices. Nasir et al. [2019], Nan et al. [2017], Nazar et al. [2019], Venticinque and Amato [2019] and Guerrero et al. [2019] discuss the stream workload in Fog.

3.6 Research Gaps in Application Architecture

Table 3 summarizes the existing concepts related to the application architecture in Fog computing. Although there is a notable number of works, some issues related to this aspect of application management are yet to be investigated. They are discussed next:

(1) The execution of one application that has a particular programming model can trigger another application with a different programming model. In such cases, the dynamic reconfiguration of Fog nodes is required. However, in existing works, only the static configuration of Fog nodes have been discussed [Goudarzi et al. 2019; Arya and Dave 2017].
(2) The varying service type of applications can affect the networking capabilities of the host Fog nodes and degrade the service time of the applications. Nevertheless, the existing

Table 3. Summary of Existing Concepts on Application Architecture

Works	Application Architecture					Works	Application Architecture				
	Program Model	Service Type	Workload Type	Interaction Method	Functional Layout		Program Model	Service Type	Workload Type	Interaction Method	Functional Layout
[Abbasi et al. 2019]		Actuation	Batch		Monolithic	[Mahmoud et al. 2018]	Dataflow		Stream		Module
[AbdElhalim et al. 2019]	Dataflow	Storage	Batch		Module	[Mass et al. 2018]	Dataflow			REST	Module
[Al-Khafajiy et al. 2019]	Dataflow, Thread		Stream			[Mazouzi et al. 2019]			Batch		Monolithic
[Anglano et al. 2019a]	Thread		Stream			[Mohamed et al. 2019]		Invocation			μ-service
[Anzanpour et al. 2019]		Invocation			Module	[Mouradian et al. 2019]	Dataflow		Batch		Module
[Arkian et al. 2017]		Display	Stream	Shared Database	Monolithic	[NAAS et al. 2018]	Dataflow	Actuation			
[Avgeris et al. 2019]			Stream		Monolithic	[Nan et al. 2017]			Stream		Monolithic
[Benamer et al. 2018]	Dataflow	Display			Module	[Nasir et al. 2019]	Map-Reduce	Display	Stream	Shared Database	Monolithic
[Binh et al. 2018]			Batch		Monolithic	[Nazar et al. 2019]	Thread		Stream		
[Chiti et al. 2019]	Thread		Batch		Monolithic	[Oma et al. 2019]	Dataflow	Actuation			Module
[Choudhari et al. 2018]		Display	Batch			[Ozeer et al. 2018]	Dataflow			Message	Module
[Concone et al. 2019]	Dataflow	Storage			Module	[Prabavathy et al. 2019]		Display			Module
[Dehnavi et al. 2019]		Actuation	Batch		Module	[Santos et al. 2019]				REST	μ-service
[Deng et al. 2016]			Stream		Monolithic	[Saurez et al. 2016]	Dataflow				Module
[Ding et al. 2019]	Dataflow	Invocation, Display	Stream		Module	[Skarlat et al. 2017]	Dataflow		Batch	Message	μ-service
[Djemai et al. 2019]	Dataflow				Module	[Su et al. 2018]			Stream		Monolithic
[Elbamby et al. 2018]		Display	Batch			[Suter et al. 2019]	Dataflow			Shared Database	Module
[Fadahunsi and Maheswaran 2019]		Invocation	Batch		Monolithic	[Tychalas and Karatza 2020]	Thread		Batch		

(Continued)

Table 3. Continued

Works	Application Architecture					Works	Application Architecture				
	Program Model	Service Type	Workload Type	Interaction Method	Functional Layout		Program Model	Service Type	Workload Type	Interaction Method	Functional Layout
[Fiandrino et al. 2019]	Dataflow	Display			Module	[Venticinque and Amato 2019]			Stream		μ-service
[Fizza et al. 2018]			Batch		Monolithic	[Verba et al. 2019]			Stream		Monolithic
[Gad-Elrab and Noaman 2020]	Dataflow				Module	[Vinueza Naranjo et al. 2018]			Stream	Message	
[Giang et al. 2020]	Dataflow	Display	Stream		Module	[Wang et al. 2019a]			Batch		Monolithic
[Guerrero et al. 2019]	Dataflow		Stream		μ-service	[Wang et al. 2019b]			Batch		Monolithic
[He et al. 2018]			Stream		Monolithic	[Wiener et al. 2019]	Dataflow			REST	μ-service
[Karamoozian et al. 2019]	Dataflow		Stream		Module	[Wu and Wang 2019]	Dataflow		Batch		
[Karatas and Korpeoglu 2019]		Storage	Stream	Shared Database	Monolithic	[Yao and Ansari 2019]			Batch		Monolithic
[Kayal and Liebeherr 2019]	Dataflow				μ-service	[Yin et al. 2018]		Display	Batch		Monolithic
[Kim et al. 2019]	Dataflow	Actuation			Module	[Yousefpour et al. 2018]			Stream		Monolithic
[Lera et al. 2019]	Dataflow				μ-service	[Yue et al. 2018]	Map-Reduce	Actuation	Batch		
[Li et al. 2018b]	Thread	Display	Batch		Monolithic	[Zeng et al. 2018]	Dataflow	Display	Stream		μ-service
[Liu et al. 2019]	Thread		Batch			[Zhou et al. 2019]			Batch		Monolithic
[Liu et al. 2017]			Stream		Monolithic	[Zhu et al. 2019]			Batch		μ-service

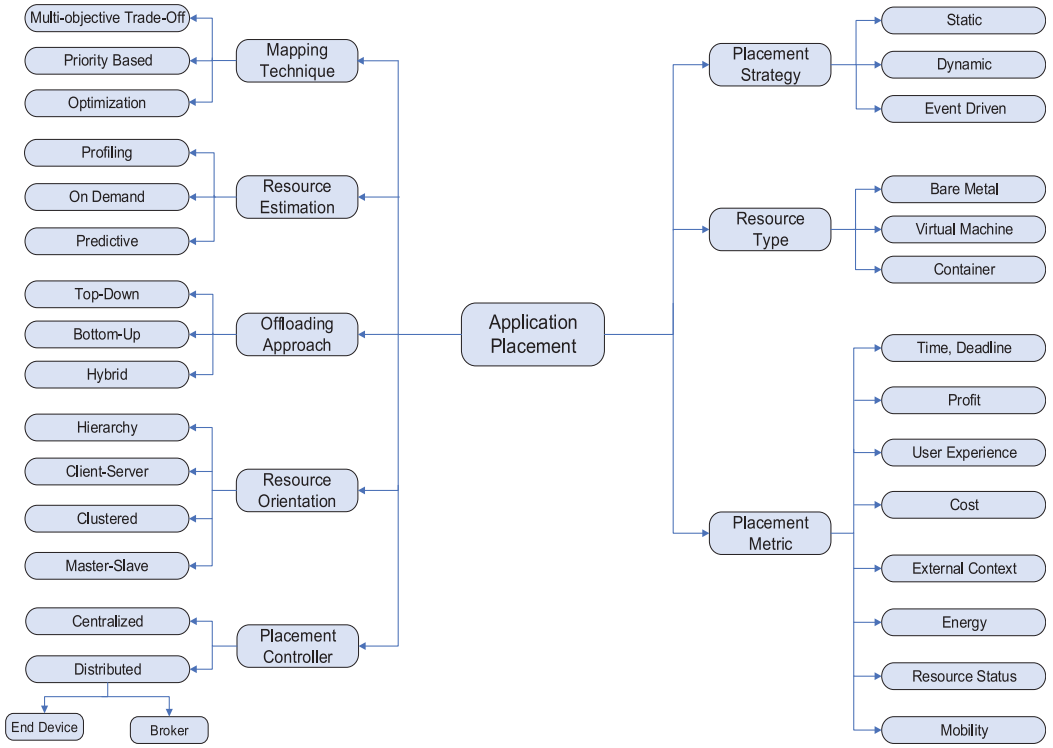

Fig. 5. Taxonomy on application placement.

approaches barely consider multiple service types of an application simultaneously while determining suitable placement options for them [Mahmud et al. 2019a].

(3) There are some research works denoting that the higher sensing frequency of IoT devices is required for better accuracy [Tuli et al. 2019]. However, they have not considered that the high streaming rate of data creates immense processing burden on the Fog nodes.

(4) Although, monolithic applications alleviate the constraints of inter-nodal communication delay, they are less modular. Conversely, the distributed applications offer scalability, but their service often is affected by the limitations of the underlying network. Although dynamic modularization of applications as per the context of a Fog network is required, current research only focuses on the fixed functional layouts [Benamer et al. 2018].

4 APPLICATION PLACEMENT

The task distribution problem in Fog computing can be solved to a great extent if the applications are placed considering the future processing commitments of the Fog nodes. Additionally, the opportunistic placement of the applications can be a potential factor to standardize Fog and Cloud integration. Moreover, while placing the applications, the resource orientation and their status are studied extensively. Such studies can play a vital role to update the application architecture dynamically and ensure proactive application maintenance. Figure 5 depicts a taxonomy of various elements relevant to application placement. Their descriptions are given in the following.

4.1 Resource Estimation

Resource estimation helps determine whether an application is compute intensive, I/O intensive or disk intensive. This information is crucial to make placement decisions with respect to the capacity constraints of Fog nodes. There are three types of resource estimation techniques in Fog: profiling, predictive and on demand.

4.1.1 Profiling. When a limited number of Fog nodes reside in a Fog environment and the specifications of requested applications remain static, the profiling technique is predominantly used to estimate the resources for an application. In this technique, an application is executed separately on each Fog node, and associated performance parameters such as processing time, propagation time and energy consumption are monitored. Based on the accumulated information, the suitable resources for the subsequent executions of the application are selected. In the work of Elbamby et al. [2018], Kim et al. [2019], Zhu et al. [2019] and Auluck et al. [2019], application profiling is discussed.

4.1.2 Predictive. In this technique, based on the past execution patterns, the appropriate resources for an application are determined. This technique is highly applicable when a Fog environment supports dynamic provisioning of its component Fog nodes and the specifications of the requested application vary. Compared to the profiling of applications, this technique is highly scalable; however, its results can be less precise. Yet profiling depends on the physical deployment, whereas prediction relies on the mathematical implications. Bhatia et al. [2019], Fang et al. [2019], Concone et al. [2019] and Zhang et al. [2019b] discuss predictive resource estimation.

4.1.3 On Demand. In this technique, resources are estimated based on the expectations of users and their instant demand. The technique differs from profiling or predictive techniques where the resource estimation depends on the application characteristics. Avgeris et al. [2019], Benblidia et al. [2019], Kim and Chung [2018] and Tuli et al. [2019] consider on-demand resource estimation.

4.2 Offloading Approach

An offloading approach helps manage the applications and their associated data as per the up-link and down-link network overhead of the host Fog nodes. In the literature, three types of offloading approaches have been found for the Fog-based applications: bottom-up, top-down and hybrid.

4.2.1 Bottom-Up. In this type of offloading, the requests regarding application services and relevant data are directly forwarded to the Fog nodes from the IoT devices or end users. This approach for Fog-based applications is highlighted in the work of Liu et al. [2017], Shah-Mansouri and Wong [2018], Mass et al. [2018] and Wazid et al. [2019].

4.2.2 Top-Down. Unlike bottom-up offloading, the top-down approach pushes the workload and programs of an application from Cloud to Fog nodes as per the requests of the end users. This technique is often applied when Fog nodes are used for content distribution. Ahn et al. [2018], Kim et al. [2019], Li et al. [2019b] and Zhou et al. [2019] consider this approach for Fog.

4.2.3 Hybrid. Apart from the aforementioned offloading approaches, Fog nodes can share application programs among themselves. During such interactions, a hybrid pattern of top-down and bottom-up offloading is followed. This approach is considered in the work of Vinueza Naranjo et al. [2018], Noura et al. [2019], Wang et al. [2019b] and Zhu et al. [2019] for Fog environments.

4.3 Resource Orientation

When different applications work collaboratively and are deployed in a distributed manner across multiple Fog nodes, the inter-nodal communication delay becomes a very important factor to meet their QoS. This delay largely depends on how the computing resources are arranged in Fog environments. There are four types of resource orientation in Fog: hierarchy, clustered, client-server and master-slave.

4.3.1 Hierarchy. In this orientation, Fog nodes are arranged in different hierarchical levels between the communication path of IoT devices and Cloud datacentres. In the lower level, the number Fog nodes is higher compared to that of the higher levels. Conversely, as the level number goes higher, the inter-nodal communication delay between the devices increases. This orientation assists in vertical scaling of the resources. In the work of Abbasi et al. [2019], AbdElhalim et al. [2019], Ali et al. [2018], Arkian et al. [2017] and Auluck et al. [2019], this type of orientation of Fog nodes is highlighted.

4.3.2 Clustered. In clustered resource orientation, all Fog nodes are directly or logically connected to each other and share the information among themselves with high-throughput communication channels. In most of the cases, the external communication, resource management and resource discovery operations within a Fog cluster are supervised by a dedicated Fog node. The clustered orientation provides more scope for horizontal scaling than the hierarchical orientation. This resource orientation is discussed in the work of Anglano et al. [2019b], Binh et al. [2018], Choudhari et al. [2018], Goudarzi et al. [2019], Kayal and Liebeherr [2019] and Liu et al. [2019].

4.3.3 Client-Server. Client-server-based resource orientation enables a set of Fog nodes to work as the servers while letting others act as the clients. The client Fog nodes request the server Fog nodes to process their forwarded data. This orientation is often regarded as a combination of clustered and hierarchical orientation. Adhikari and Gianey [2019], Alnoman and Anpalagan [2018], Alrawais et al. [2017], Arya and Dave [2017], Avgeris et al. [2019] and Chiti et al. [2019] discuss the client-server orientation of Fog resources.

4.3.4 Master-Slave. In this orientation, a master Fog node distributes the data processing responsibilities to other slave Fog nodes and explicitly manages their operations. After receiving the service outcomes from the slave nodes, the master node accumulates them and forwards the final results to the destination as per the service type. This orientation is more efficient in distributing the computing responsibilities than the client-server orientation. In the work of Fahs and Pierre [2019], Benblidia et al. [2019], He et al. [2018], Santos et al. [2019], Nasir et al. [2019] and Tuli et al. [2019], the master-slave resource orientation is considered.

4.4 Placement Controller

The placement controller defines the logical location of an entity that manages the overall application management operations in Fog computing. Furthermore, it assists in estimating the waiting time from requesting to placing an application in the Fog environment which consequently drives the overall performance of the IoT-enabled CPSs. There are two types of placement controllers in Fog: centralized and distributed.

4.4.1 Centralized. This type of placement controller is located in a commonly accessible place by the Fog nodes and poses a global view of the Fog environment. Generally, the centralized placement controller is hosted in Cloud datacentres and supervises the Fog nodes residing at the edge network. Concone et al. [2019], Kim et al. [2019], Li et al. [2019b], Santos et al. [2019], Mishra et al. [2018] and Wang et al. [2019a] discuss the centralized placement controllers of Fog environments.

4.4.2 Distributed. Unlike the centralized controller, the distributed placement controllers manage the Fog nodes based on a local view of the Fog environments. They are classified into two categories: end device and broker.

- *End device*: In this type of distributed placement controller, the IoT devices and the Fog nodes not only perform their predefined responsibilities such as data sensing and data processing, but also take the application management decisions for other Fog nodes. Li et al. [2019a], Liu et al. [2017], Liu et al. [2019], Vinueza Naranjo et al. [2018] and Rahbari and Nickray [2019] consider end devices as distributed placement controller for Fog environments.
- *Broker*: In contrast to end devices, this type of controller is considered to be dedicated for application management operations in Fog environments. The brokers reside in proximity of the Fog nodes and interact with the external entities on behalf of the Fog nodes and vice versa. Chen et al. [2019], Filiposka et al. [2019], He et al. [2018], Jiang et al. [2019], Skarlat et al. [2017] and Sun et al. [2019] discuss broker-based placement controllers in Fog environments.

4.5 Mapping Technique

Based on different parameters, an application placement policy provides the mapping of the applications with respect to the Fog nodes and their virtualized instances. The complexity of the adopted mapping technique defines the runtime of the policy which consequently denotes its responsiveness. Three different types of mapping techniques are commonly used in Fog computing: priority based, optimization and multi-objective trade-off.

4.5.1 Priority Based. This technique prioritizes an application placement on particular Fog node or virtualized instances. Generally, heuristic approaches such as best fit and first fit are commonly used for prioritization of the applications. In the work of Alnoman and Anpalagan [2018], Anzanpour et al. [2019], Arya and Dave [2017], Li et al. [2018b], Santos et al. [2019] and Saurez et al. [2016], prioritization is highlighted as the mapping technique.

4.5.2 Optimization. This technique either maximizes or minimizes one particular objective function while placing the applications in Fog environments. Although optimization provides the best mathematical solution to a problem, this technique takes more time to operate than prioritization. Different types of optimization approaches such as linear, non-linear and constrained are widely studied in Fog computing. Djemai et al. [2019], Elbamby et al. [2018], Benamer et al. [2018], Bhatia et al. [2019], Dehnavi et al. [2019] and Auluck et al. [2019] discuss this mapping technique for Fog.

4.5.3 Multi-Objective Trade-Off. Unlike optimization, multi-objective trade-off can maximize or minimize two or more objectives, such as time-energy, time-cost and cost-QoE, simultaneously while placing the applications. Different meta-heuristic approaches including particle swarm, evolutionary algorithms, game theory and multi-objective optimization are used for making a trade-off among various application placement metrics. AbdElhalim et al. [2019], Fahs and Pierre [2019], Binh et al. [2018], Fang et al. [2019] and Mishra et al. [2018] consider trade-off for placing the applications.

4.6 Placement Strategy

The iterative execution of applications in Fog environments depends on the arrival of their inputs, or data processing life cycle. The placement algorithms need to consider these issues so that they can detect suitable hosts for different applications. The placement strategy helps define how frequently the placement algorithms are required to be executed for subsequent execution of an

application. There are three types of placement strategies for Fog computing: static, dynamic and event driven.

4.6.1 Static. In this strategy, the placement algorithm is executed once for each application, and at the host, the application is kept running. Inputs of an application are directly sent to its host from the IoT devices as the processing destination remains the same for all of them. Adhikari and Gianey [2019], Ali et al. [2018], Alrawais et al. [2017], Dehnavi et al. [2019], Deng et al. [2016] and Li et al. [2019a] discuss the static application placement strategy in Fog computing environments.

4.6.2 Dynamic. In Fog, an application can have multiple replicas running or the application can terminate by processing only a single input. For both cases, the placement algorithm requires texecution for each arrival of the inputs to determine where to schedule them or where to execute the application next. Such dynamic of placement strategy is highlighted in the work of Abbasi et al. [2019], Ahn et al. [2018], Kayal and Liebeherr [2019], Lee et al. [2019] and Zhang et al. [2019a].

4.6.3 Event Driven. An application often requires relocation from one host to another. This relocation can occur because of the mobility of the sources and destinations, preemption, Fog node consolidation or service migration. In such cases, after initial placement, further scheduling of applications is conducted occasionally based on the occurrence of the event. The event-driven strategy for Fog-based application placement is considered in the work of Fadahunsi and Maheswaran [2019], Benamer et al. [2018], Li et al. [2018b], Puliafito et al. [2018] and Zhu et al. [2019].

4.7 Resource Type

Fog nodes contain necessary resources such as processing cores, memory and bandwidth to assist in the execution of applications. They can be virtualized to support multi-tenancy on the physical resources. The resource type denotes the internal features of the host of the applications, which helps in validating the scope of dynamic allocation of resources during application runtime. Three different types of resources are discussed in Fog computing: bare metal, virtual machine and container.

4.7.1 Bare Metal. In this type of resource, applications are directly placed to the Fog nodes without any virtualization. Applications access the physical hardware of Fog nodes through the host operating system. Such a type of resources can support multi-tenancy without explicit isolation of the application execution unit. Elbamby et al. [2018], Fadahunsi and Maheswaran [2019], Benamer et al. [2018], Benblidia et al. [2019], Chiti et al. [2019] and Nan et al. [2017] highlight bare metal as resource type of Fog nodes.

4.7.2 Virtual Machine. In contrast to bare metal resources, virtual machines exploit hardware-level virtualization so that multiple operating systems can run independently on a single Fog node. They run on top of an abstraction layer named the *hypervisor* that enables the sharing of hardware among different virtual machines. Avgeris et al. [2019], Fan et al. [2017], Chen et al. [2019], Filiposka et al. [2019] and Mishra et al. [2018] consider application placement in virtual machines.

4.7.3 Container. This type of virtualized resource is lightweight compared to virtual machines and offers operating-level virtualization. In opposition to bare metals, containers isolate processes with required application packages and are highly portable across multiple Fog nodes. Containers are used in the work of Fahs and Pierre [2019], Kim and Chung [2018], Luo et al. [2019], Santos et al. [2019], Pallewatta et al. [2019] and Wiener et al. [2019] for application placement in Fog environments.

4.8 Placement Metric

The main intention of placing applications in Fog can vary according to the requirements of users, service providers and physical environments. The placement metric refers to the parameters that set the objectives of application placement in Fog environments. A wide variety of placement metrics are noted in Fog computing. They are described next.

4.8.1 Time and Deadline. This placement metric signifies the aim of minimizing application service delivery time and meeting the specified deadline. While setting this metric, the computation time, data propagation time and node deployment time are also considered. Alnoman and Anpalagan [2018], Elbamby et al. [2018], Benamer et al. [2018], Brogi and Forti [2017], Huang et al. [2019] and Lee et al. [2019] discuss time as the placement metric in Fog.

4.8.2 Profit. Service providers benefit when the applications are deployed in Fog with a view to maximizing their profit and revenue. The intention of making profit often leads the providers to offer application execution in Fog as a utility. Anglano et al. [2019b], Fan et al. [2017], Wang et al. [2019a], Mahmud et al. [2020] and Zhou et al. [2019] consider profit as the placement metric.

4.8.3 User Experience. Service requirements of users and their affordability level can change over time. If these issues are not met during application placement, the user experience can degrade. This event could also cause users to resist executing applications through Fog computing in the future. Arya and Dave [2017], Benblidia et al. [2019], Chen et al. [2019], Mahmud et al. [2019b] and Shah-Mansouri and Wong [2018] highlight the user experience as the placement metric.

4.8.4 Cost. Different monetary costs, such as infrastructure deployment cost, operational cost and instance rental cost, are associated with Fog computing. Cost as a placement metric refers to its minimization during application placement in Fog. This metric is considered in the work of Arkian et al. [2017], Ding et al. [2019], Binh et al. [2018], Choudhari et al. [2018], He et al. [2018] and Auluck et al. [2019] to place the applications in Fog.

4.8.5 External Context. Several external parameters, including the relinquish rate and the activity of users, reliability of Fog nodes, popularity of application services, data size and sensing frequency of IoT devices, drive the decision of application placement in Fog. Alrawais et al. [2017], Anglano et al. [2019a], Anzanpour et al. [2019], Fang et al. [2019], Concone et al. [2019] and Zhu et al. [2019] consider such external contexts while placing applications in Fog.

4.8.6 Energy. Fog nodes can utilize both renewable and non-renewable energy to operate. However, the energy consumption of Fog nodes is subject to environmental- and supply-demand-related issues. AbdElhalim et al. [2019], Adhikari and Gianey [2019], Alli and Alam [2019], Djemai et al. [2019] and Jiang et al. [2019] highlight energy as a placement metric for the application.

4.8.7 Resource Status. Fog nodes are widely heterogeneous in terms of their processing power, networking interfaces, storage capacity and operational platform. Assessment of these status parameters is very important to efficiently manage the applications over them. Abbasi et al. [2019], Ali et al. [2018], Alli and Alam [2019], Arya and Dave [2017], Avgeris et al. [2019] and Dehnavi et al. [2019] give higher preference to resource status while placing the applications in Fog environments.

4.8.8 Mobility. In the context of Fog computing, both IoT devices and Fog nodes can move from one location to another frequently. This feature of Fog computing affects the service delivery time of the applications, which consequently leads Fog nodes to migrate the application execution among themselves. Recognizing these issues, mobility is considered in the work of Chen et al.

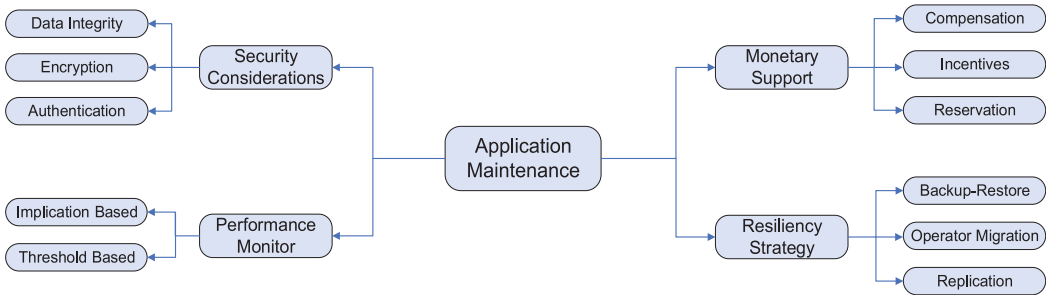

Fig. 6. Taxonomy on application maintenance.

[2019], Filiposka et al. [2019], Mass et al. [2018], Niu et al. [2018], Suarez et al. [2016] and Zhu et al. [2019] for Fog computing.

4.9 Research Gaps in Application Placement

Table 4 summarizes the existing application placement techniques in Fog computing. Although an extensive amount of research has been conducted on this aspect of application management, there are still some gaps, which are discussed next:

(1) For remote areas, many research works suggest using renewable power sources to run the Fog nodes, as grid-based energy is costly to facilitate [Nan et al. 2017]. However, few of them consider that the availability of renewable energy is subjected to uncertainty and environmental context and take the required measures to solve the problem.
(2) The distribution of application placement tasks across multiple entities can reduce management overhead. However, existing works have not considered the elevation in decision-making time that can occur due to such distribution [Wang et al. 2019b; Pallewatta et al. 2019].
(3) Most existing works prefer Cloud to place applications when there is no resource available in the corresponding Fog infrastructure [Mahmud et al. 2020]. However, they have not discussed the confederation of Fog infrastructure owned by different service providers. As a consequence, the scope of performance improvement lessens and the providers fail to harness the monetary benefits.
(4) The consolidation of Fog nodes can save energy. However, this operation can alter the topology and orientation of Fog resources and affect the collaborative execution of applications. Despite such impact, current research barely looks into this issue [Chen et al. 2019].

5 APPLICATION MAINTENANCE

Robust application maintenance operations are required to secure the access of all legitimate users to application outcomes in Fog environments. Additionally, if these operations are conducted in both a proactive and reactive manner, the uncertain failure of Fog nodes and the performance degradation of applications can be mitigated to a certain extent. The demand of application maintenance can also trigger the necessity to introduce additional features such as check pointing and partitioning to the functional layout of application architecture. Moreover, the requirements of operator migration can initiate event-driven application placement. Figure 6 illustrates a taxonomy of different elements of application maintenance. In the following sections, they are discussed in detail.

Table 4. Summary of Existing Application Placement Techniques

Works	Application Placement							
	Mapping Technique	Resource Estimation	Offloading Approach	Resource Orientation	Placement Controller	Placement Strategy	Resource Type	Placement Metric
[Abbasi et al. 2019]				Hierarchy	Centralized	Dynamic	VM	Time, Resource
[AbdElhalim et al. 2019]	Trade-Off		Bottom-Up	Hierarchy	Broker		Bare Metal	Time, Energy
[Adhikari et al. 2019]	Priority		Hybrid	Hierarchy	End Device	Static	Bare Metal	Time, Resource
[Adhikari and Gianey 2019]	Optimization		Bottom-Up	Client-Server	Broker	Static	Bare Metal	Time, Energy
[Ahn et al. 2018]	Optimization		Top-Down			Dynamic		QoE
[Ali et al. 2018]	Optimization			Hierarchy	End Device	Static	Bare Metal	Time, Resource
[Al-Khafajiy et al. 2019]	Priority		Hybrid	Cluster	End Device	Dynamic	Bare Metal	Time, Resource
[Alli and Alam 2019]			Bottom-Up	Hierarchy	End Device	Dynamic	VM	Energy, Resource
[Alnoman and Anpalagan 2018]	Priority			Client-Server	Centralized	Dynamic	VM	Time
[Alrawais et al. 2017]			Top-Down	Client-Server	Centralized	Static	Bare Metal	Context
[Anglano et al. 2019b]	Optimization			Cluster	Centralized	Static	VM	Profit
[Anglano et al. 2019a]			Bottom-Up		Broker	Dynamic	VM	Context
[Anzanpour et al. 2019]	Priority	Predictive				Dynamic	Bare Metal	Context, Energy
[Arkian et al. 2017]	Optimization			Hierarchy		Event driven	VM	Cost
[Arya and Dave 2017]	Priority		Bottom-Up	Client-Server	Broker	Dynamic	VM	QoE, Resource
[Avgeris et al. 2019]	Optimization	On Demand	Hybrid	Client-Server	End Device, Broker		VM	Time, Resource
[Auluck et al. 2019]	Optimization	Profiling		Hierarchy	Broker		Bare Metal	Cost
[Behera et al. 2020]		Predictive		Hierarchy		Dynamic	VM	Resource
[Bellavista et al. 2019]			Hybrid		Broker	Event Driven	Container	Energy, Context
[Benamer et al. 2018]	Optimization			Hierarchy		Event Driven	Bare Metal	Time
[Benblidia et al. 2019]	Priority	On Demand		Master-Slave	Broker		Bare Metal	Time, QoE, Energy
[Bhatia et al. 2019]	Optimization	Predictive				Event Driven	Bare Metal	Resource
[Binh et al. 2018]	Trade-Off			Cluster			Bare Metal	Time, Cost
[Brogi and Forti 2017]	Priority	Profiling		Hierarchy		Static	Bare Metal	Time
[Chen et al. 2019]	Optimization				Broker	Event-Driven	VM	QoE, Resource, Mobility

(Continued)

Table 4. Continued

Works	Application Placement							
	Mapping Technique	Resource Estimation	Offloading Approach	Resource Orientation	Placement Controller	Placement Strategy	Resource Type	Placement Metric
[Chiti et al. 2019]	Optimization		Bottom-Up	Client-Server	End Device	Static	Bare Metal	Time, Energy
[Choudhari et al. 2018]	Priority			Cluster	Centralized	Static	VM	Time, Cost
[Charântola et al. 2019]	Priority		Bottom-Up	Hierarchy	End Device	Event Driven	Bare Metal	Time, Mobility
[Concone et al. 2019]		Predictive		Hierarchy	Centralized	Event Driven	Bare Metal	Context
[de Souza Toniolli and Jaumard 2019]	Priority			Hierarchy	Broker	Static		Time, Cost
[Dehnavi et al. 2019]	Optimization			Hierarchy		Static		Time, Resource
[Deng et al. 2016]				Hierarchy		Static	Bare Metal	Time, Energy
[Ding et al. 2019]	Optimization			Client-Server	End Device	Static		Time, Cost
[Djemai et al. 2019]	Optimization			Hierarchy			Bare Metal	Time, Energy
[Du et al. 2018]	Optimization		Bottom-Up	Client-Server	Broker	Static		Time, Energy
[Elbamby et al. 2018]	Optimization	Profiling	Bottom-Up	Client-Server		Static	Bare Metal	Time
[Fadahunsi and Maheswaran 2019]				Hierarchy	Broker	Event Driven	Bare Metal	Time, Resource
[Fahs and Pierre 2019]	Trade-Off		Bottom-Up	Master-Slave	End Device		Container	Resource
[Fan et al. 2017]	Optimization		Hybrid		Centralized	Dynamic	VM	Time, Profit
[Fang et al. 2019]	Trade-Off	Predictive		Client-Server	End Device			Context, Energy, Resource
[Farhat et al. 2019]	Priority	On Demand		Cluster	Centralized			Resource
[Filiposka et al. 2019]	Optimization			Hierarchy	Broker	Event Driven	VM	Mobility
[Fizza et al. 2018]				Hierarchy		Dynamic	Bare Metal	Time, Context
[Gazori et al. 2019]		Predictive		Hierarchy	Broker		VM	Time, Cost
[Ghosh et al. 2019]				Hierarchy	Centralized		Bare Metal	Time, Mobility
[Guerrero et al. 2019]	Priority	Predictive		Hierarchy	End Device	Event Driven		Context
[Gad-Elrab and Noaman 2020]	Trade-Off			Cluster	Broker		VM	Time, Cost, Energy
[He et al. 2018]				Master-Slave	Broker		Bare Metal	Cost, Resource
[He et al. 2019]	Optimization	On Demand				Dynamic	Container	Mobility
[Huang et al. 2019]	Optimization			Hierarchy			Bare Metal	Time
[Jamil et al. 2019]	Priority		Hybrid	Hierarchy	Broker			Time, Energy

(Continued)

Table 4. Continued

Works	Application Placement							
	Mapping Technique	Resource Estimation	Offloading Approach	Resource Orientation	Placement Controller	Placement Strategy	Resource Type	Placement Metric
[Jiang et al. 2019]	Priority		Bottom-Up	Client-Server	Broker	Dynamic	Bare Metal	Time, Energy
[Josilo and Dan 2019]	Optimization		Bottom-Up	Client-Server	End Device	Dynamic	Bare Metal	Time
[Kamal et al. 2019]	Optimization			Hierarchy		Static	VM	Time, Cost
[Karamoozian et al. 2019]	Optimization			Cluster		Static	Bare Metal	Time, Cost
[Karatas and Korpeoglu 2019]	Optimization	Profiling		Hierarchy	Broker	Static	Bare Metal	Time, Cost
[Kayal and Liebeherr 2019]	Optimization			Cluster	End Device	Dynamic	Bare Metal	Cost, Energy
[Kim et al. 2019]	Optimization	Profiling	Top-Down		Centralized	Event Driven		Time
[Kim and Chung 2018]		On Demand			Broker		Container	Cost
[Lee et al. 2019]	Optimization		Hybrid	Cluster	End Device	Dynamic	Bare Metal	Time
[Lera et al. 2019]	Priority			Cluster		Dynamic	Bare Metal	Resource
[Li et al. 2019b]	Optimization		Top-Down	Hierarchy	Centralized	Static		Cost
[Li et al. 2018b]	Priority		Hybrid	Cluster	End Device	Event Driven		Resource
[Li et al. 2019c]	Trade-Off			Hierarchy	End Device		Bare Metal	Time, Energy
[Li et al. 2019a]			Bottom-Up	Hierarchy	End Device	Static	VM	Time
[Liu et al. 2019]	Optimization		Hybrid	Cluster	End Device	Static	Bare Metal	Time
[Liu et al. 2017]	Optimization		Bottom-Up		End Device		Bare Metal	Time, Cost, Energy
[Luo et al. 2019]				Hierarchy	End Device		Container	Time, Resource
[Mahmoud et al. 2018]				Hierarchy	End Device		VM	Energy
[Mahmud et al. 2018c]	Priority		Hybrid	Hierarchy, Cluster	End Device	Static	Bare Metal	Time, Resource
[Mahmud et al. 2019]	Optimization			Cluster	Centralized, Broker	Static	VM, Container	Context, Resource
[Mahmud et al. 2019a]	Priority		Hybrid	Hierarchy, Cluster	Broker	Dynamic	VM, Container	Time, Context
[Mahmud et al. 2019b]	Optimization	Profiling		Hierarchy	Broker	Static	Bare Metal	QoE
[Mahmud et al. 2020]	Priority		Hybrid	Cluster	Broker	Dynamic	VM, Container	Profit, Cost

(Continued)

Table 4. Continued

Works	Application Placement							
	Mapping Technique	Resource Estimation	Offloading Approach	Resource Orientation	Placement Controller	Placement Strategy	Resource Type	Placement Metric
[Mass et al. 2018]			Bottom-Up	Client-Server	End Device	Dynamic	VM, Container	Resource, Mobility
[Mazouzi et al. 2019]	Priority			Client-Server	End Device	Dynamic		Time
[Meixner et al. 2019]	Optimization		Top-Down		Centralized		Container	Context
[Melnik et al. 2019]				Client-Server	End Device		Bare Metal	Context
[Mishra et al. 2018]	Trade-Off			Client-Server	Centralized	Dynamic	VM	Time, Energy
[Mishra et al. 2019]	Optimization			Hierarchy	Broker			Time, Energy
[Mtshali et al. 2019]	Optimization				End Device	Static	VM, Container	Energy, Resource
[NAAS et al. 2018]	Optimization			Client-Server			Bare Metal	Time
[Nair and Somasundaram 2019]	Priority	Profiling		Client-Server	Broker		VM	Time
[Nan et al. 2017]	Trade-Off		Bottom-Up	Hierarchy	Broker		Bare Metal	Time, Cost, Energy
[Nasir et al. 2019]				Master-Slave	Centralized		Bare Metal	Energy
[Nazar et al. 2019]	Priority			Hierarchy	End Device		VM	Energy
[Niu et al. 2018]	Optimization			Client-Server	Centralized	Dynamic	Bare Metal	Mobility
[Noura et al. 2019]			Hybrid	Cluster	End Device		Bare Metal	
[Puliafito et al. 2018]					Centralized	Event Driven	Container	Mobility
[Rahbari and Nickray 2019]	Priority			Hierarchy	End Device			Resource
[Ramli et al. 2019]	Optimization		Bottom-Up	Master-Slave	Broker		Bare Metal	Cost
[Santos et al. 2019]	Priority			Master-Slave	Centralized		Container	Resource
[Saurez et al. 2016]	Priority			Hierarchy			Container	Mobility
[Shah-Mansouri and Wong 2018]	Optimization		Bottom-Up	Hierarchy	End Device	Dynamic	VM	QoE
[Sharma and Saini 2019]	Priority			Hierarchy, Cluster	Broker		Bare Metal	Time, Resource
[Shooshtarian et al. 2019]	Priority			Cluster	End Device		Bare Metal	Context
[Singh et al. 2019]	Priority		Top-Down		Centralized		VM	Energy, Resource

(Continued)

Table 4. Continued

Works	Application Placement							
	Mapping Technique	Resource Estimation	Offloading Approach	Resource Orientation	Placement Controller	Placement Strategy	Resource Type	Placement Metric
[Singh and Auluck 2019]	Priority			Hierarchy			Bare Metal	Resource
[Skarlat et al. 2017]	Priority			Cluster	Broker		VM, Container	Time, Resource
[Su et al. 2018]	Optimization			Cluster	End Device			Time, Cost
[Sun et al. 2019]	Optimization			Hierarchy	Broker	Dynamic		Time, Energy
[Suter et al. 2019]	Priority			Cluster			Bare Metal	Cost
[Venticinque and Amato 2019]	Optimization	Profiling		Cluster	Broker			Time, Resource
[Verba et al. 2019]		Profiling		Cluster	Broker	Dynamic		Time, Resource
[Vinueza Naranjo et al. 2018]	Optimization		Hybrid	Hierarchy, Cluster	End Device		Container	Energy
[Wang et al. 2019]	Optimization			Cluster			Bare Metal	Time
[Vu et al. 2019]	Optimization		Bottom-Up		Centralized		Bare Metal	Time, Energy
[Tuli et al. 2019]		On Demand	Bottom-Up	Master-Slave	Broker	Dynamic	Bare Metal	
[Tychalas and Karatza 2020]	Priority				Broker		VM, Container	Resource
[Wang et al. 2019a]	Optimization		Bottom-Up	Hierarchy	Centralized	Event Driven		Profit, Mobility
[Wang et al. 2019]			Bottom-Up	Client-Server	Broker	Event Driven		Time, Cost
[Wang et al. 2018]	Optimization			Hierarchy	Broker			Time, Context
[Wang et al. 2019b]	Trade-Off		Hybrid	Client-Server	End Device			Time, Energy
[Wazid et al. 2019]			Bottom-Up	Client-Server	Broker		Bare Metal	
[Wiener et al. 2019]				Master-Slave	End Device	Event Driven	Container	Resource
[Wu and Wang 2019]	Priority			Hierarchy		Static	Bare Metal	Time
[Yao et al. 2019]			Top-Down	Client-Server	Centralized	Dynamic	Container	
[Xiao and Krunz 2018]	Trade-Off		Hybrid		End Device	Static	Bare Metal	Time, Energy

(Continued)

Table 4. Continued

Works	Application Placement							
	Mapping Technique	Resource Estimation	Offloading Approach	Resource Orientation	Placement Controller	Placement Strategy	Resource Type	Placement Metric
[Yao and Ansari 2019]	Optimization	On Demand		Client-Server			VM	Time, Cost
[Yin et al. 2018]	Optimization			Client-Server	End Device		Container	Time
[Yousefpour et al. 2018]	Priority		Hybrid		Centralized	Static		Context, Resource
[Yue et al. 2018]				Cluster		Static	Bare Metal	Resource
[Zeng et al. 2018]	Optimization			Cluster			Bare Metal	Energy, Resource
[Zeng et al. 2018]			Hybrid	Client-Server			Bare Metal	Time, Cost, Energy
[Zhang et al. 2019b]	Priority	Predictive		Cluster	Centralized	Dynamic		Energy
[Zhang et al. 2019a]	Optimization			Client-Server	End Device	Dynamic	Bare Metal	Time
[Zhao et al. 2019]	Optimization			Cluster		Event Driven		Mobility
[Zhao et al. 2018]	Optimization		Top-Down	Hierarchy	Centralized	Dynamic	Bare Metal	Resource
[Zheng et al. 2019a]			Hybrid	Hierarchy	End Device		Bare Metal	Energy
[Zheng et al. 2019b]	Optimization		Hybrid	Client-Server	End Device		Bare Metal	Energy
[Zhou et al. 2019]	Optimization		Top-Down	Client-Server	Centralized			Time, Cost
[Zhou et al. 2019]	Priority		Top-Down	Master-Slave	Centralized	Dynamic		Profit
[Zhu et al. 2019]	Trade-Off	Profiling	Hybrid	Client-Server	Broker	Event Driven	Container	Time, Context, Mobility

5.1 Security Considerations

Fog computing functions at the edge network. The attackers can access the Fog infrastructure easily and resist the smooth execution of applications by generating security threats including information impairment, identity disclosure, replay and denial of service (DoS) attacks. Therefore, Fog infrastructure is required to specify the security while executing the applications. Three different types of security are widely considered in Fog computing: data integrity, encryption and authentication.

5.1.1 Data Integrity. In some cases, various sensitive data and their processing outcomes such as patient's blood group and electronic health reports are consistently analysed by different parties, including hospitals and insurance companies. Fog computing offers easy access to these data and information with a guarantee of no alteration. Such initiatives help data integrity of applications in Fog environments. Tuli et al. [2019], Rahbari and Nickray [2019] and Suarez et al. [2016] discuss data integrity for Fog computing.

5.1.2 Encryption. In Fog computing, extensive data exchange operations are conducted between the IoT devices, Fog nodes and Cloud datacentres. Encryption not only hides the details of the transferred data but also protects the credentials of legitimate users to access the Fog resources. In the work of Alrawais et al. [2017], Fiandrino et al. [2019], Concone et al. [2019], Noura et al. [2019] and Su et al. [2018], encryption for Fog computing is discussed.

5.1.3 Authentication. Authentication helps identify the legitimate user of application services and Fog resources. Sometimes authentication is robustly applied at the receiver side to control the data access rate of different entities. Concone et al. [2019], Luo et al. [2019], Vinueza Naranjo et al. [2018], Rahbari and Nickray [2019] and Yao et al. [2019] consider authentication as a security measure for application maintenance.

5.2 Performance Monitor

During application runtime, a continuous monitoring of resources is required to maintain the execution flow of the applications at the desired level. Two different types of performance monitoring techniques are widely used in Fog computing: implication based and threshold based.

5.2.1 Implication Based. In this technique, the current synthesis of application and resources are implied to predict future performance trends. Implication-based performance monitoring assists in maintaining the execution of the applications in a proactive manner. Ahn et al. [2018], Avgeris et al. [2019], Bhatia et al. [2019], Li et al. [2019a] and Wang et al. [2019a] discuss implication-based performance monitoring for Fog.

5.2.2 Threshold Based. When an application is executed in a Fog node, its key performance indicators, such as processor usage and memory consumption, are compared with a dynamically set or predefined threshold value. If the state of the indicator does not match the threshold, necessary decisions are made to continue execution of the application in Fog environments. Unlike the implications, this approach helps in reactive application maintenance. Arya and Dave [2017], Elbamby et al. [2018], Fadahunsi and Maheswaran [2019], Dehnavi et al. [2019] and Tychalas and Karatza [2020] discuss threshold-based performance monitoring.

5.3 Monetary Support

Monetary support defines how Fog service providers nurture the economic interests of users while executing their requested applications. At the same time, the monetary support helps providers

attain the trust of the users that controls the relinquish rate. Three types of monetary support are found in Fog computing: compensation, incentives and reservation.

5.3.1 Compensation. Due to some uncertain events, such as node failure and security threats, the service-level agreement between users and providers can be violated. In these cases, the provider offers compensation to users so that they can rely on Fog-based execution of their requested applications. Compensation also helps providers quantify the aim of reducing service violations. Mahmud et al. [2020], Battula et al. [2019], Yu et al. [2017] and Kim and Chung [2018] highlight the compensation approach for Fog computing.

5.3.2 Incentives. At the edge network, there exist different idle computing resources that can be used as potential Fog nodes. Fog service providers often harness them to meet the additional demand of users by providing incentives to the owner of the resources. In some cases, for relaxing stringent requirements, the users also receive incentives from the service providers. He et al. [2018], Josilo and Dan [2019], Li et al. [2019b], Zeng et al. [2018] and Zhou et al. [2019] consider incentives as monetary support to maintain application execution in Fog.

5.3.3 Reservation. Reservation assists users in provisioning a certain number of applications at any given time on fixed charges despite the current load on the Fog infrastructure. In this case, Fog computation is referred to as a subscription-based utility. Fizza et al. [2018], Aazam and Huh [2015], Huang and Xu [2016] and Mahmud et al. [2019a] discuss reservation as a monetary support for application users in Fog computing.

5.4 Resiliency Strategy

Resiliency strategy denotes how Fog computing continues application execution after the occurrences of uncertain events and failures. Resiliency strategies assist in enhancing the reliability of the system. Three types of resilience strategies are widely adopted in Fog computing: backup-restore, replication and operator migration.

5.4.1 Backup-Restore. In this strategy, the intermediate results and different execution phase of an application are continuously stored so that the execution of the application can be re-initiated soon after the anomaly occurs. This operation is performed by setting some check points and temporary storage operations during the application runtime. Backup-restore is feasible when a one-to-one interaction happens between the users and the applications. [Elbamby et al. 2018], [Mohamed et al. 2019], [Noura et al. 2019] and [Ozeer et al. 2018] discuss backup-restore as a resiliency strategy.

5.4.2 Replication. In this resiliency strategy, multiple instances of an application are run across different Fog nodes. Replication ensures the satisfaction of user requests through an application even after the failure of its several instances. Unlike backup-restore, replication is well-suited for supporting the one-to-many interactions between the users and the applications. Replication for Fog is discussed in [Anglano et al. 2019a], [Fadahunsi and Maheswaran 2019], [Fahs and Pierre 2019], [Dehnavi et al. 2019] and [Wang et al. 2019].

5.4.3 Operator Migration. In case of node failure or mobility of the requesting entities, the execution of applications is often shifted from one node to another Fog node. Such migration of operators happens dynamically so that application execution continues without interrupting the user experience. As a resiliency strategy, operator migration differs from the backup-restore and replication because of its ease of scalability. [Chen et al. 2019], [Filiposka et al. 2019], [Guerrero et al. 2019], [Nair and Somasundaram 2019] and [Yao et al. 2019] discuss operator migration in Fog.

5.5 Research Gaps in Application Maintenance

Table 5 summarizes the existing application maintenance operations in Fog computing. Although extensive research initiatives have been taken, there are still some issues that require to be investigated for efficient application maintenance in Fog computing, which are discussed next:

(1) In many research works, compute intensive algorithms are used to secure the data transmission within Fog environments [Yanez et al. 2020]. However, they have not considered that heavyweight security techniques slow down the legitimate access to application services and resources.
(2) Streaming applications require reserved resources so that their processing destinations do not change very frequently. However, while making such arrangements, the existing works barely consider the waiting time of the other less-interactive applications [Aazam and Huh 2015].
(3) There are a good number of research works that mention check pointing and replication as the means of fault tolerance in Fog computing [Oma et al. 2019]. However, to deal with the scarcity of resources, they have not provided any concrete model that can dynamically tune the frequency of check points within an application and change the number of replications in Fog environments.

6 A PERSPECTIVE APPLICATION MANAGEMENT FRAMEWORK FOR FOG

In some cases, the existing application management strategies for Fog computing can be used to solve the classical research problems in mobile distributed computing [Hassan et al. 2015]. This perception is often used to bind the concept of MCC, MEC and Fog computing to a single body. However, these computing paradigms differ from each other in both architecture and operations. For example, MCC facilitates users to offload the compute- and data-intensive mobile applications to Cloud for execution and overcomes the limitations of smart phones in terms of energy, storage and computation. MCC is designed with an additional computing layer of Cloudlets in between the smart phones and Cloud datacentres to offer latency-sensitive mobile application services [Mahmud et al. 2016]. Conversely, in MEC, a virtualized server is deployed at the cellular base station to ensure flexible and rapid initiation of cellular services for the users. MEC offers real-time access to radio network information to endorse Tactile Internet, interactive gaming and virtual reality applications through 5G [Afrin et al. 2017]. In comparison to MCC and MEC, Fog computing mostly deals with the IoT-driven use cases at the edge network. Rather than harnessing virtual cellular base stations or Cloudlets, Fog aims at building a Cloud of Things in the proximity through dedicated or ad hoc networking. In Fog, the computing infrastructure is multi-tiered, whereas for MEC and MCC, it is two and three tiered, respectively [Klas 2015]. Most importantly, Fog computing provides a scope to distribute the application management operations across different tiers of the computing infrastructure; however, in other paradigms, this scope is very limited. To illustrate this feature of Fog, a perspective framework is depicted in Figure 7. At each infrastructure level, the components of this framework perform some specific operations related to application management. They are summarized in the following.

6.1 CPS Level

At the CPS level, IoT-enabled CPSs reside that request the Fog environment for the services of different applications. In this case, the IoT Application Brokers (IABs) deployed at the Fog gateway level assist the CPSs. While making the requests, the CPSs forward the specification of the applications including the workload type, the frequency of incoming data, the form of application services and the QoS requirements such as service deadline, budget and user expectations to

Table 5. Summary of Existing Application Maintenance Operations

Works	Application Maintenance			
	Security Considerations	Performance Monitor	Monetary Support	Resiliency Strategy
[Aazam and Huh 2015]			Reservation	
[Ahn et al. 2018]		Implication		
[Alrawais et al. 2017]	Encryption			
[Anglano et al. 2019a]				Replication
[Arya and Dave 2017]		Threshold		
[Avgeris et al. 2019]		Implication		
[Bhatia et al. 2019]		Implication		
[Chen et al. 2019]				Migration
[Dehnavi et al. 2019]		Threshold		Replication
[Elbamby et al. 2018]		Threshold		Backup-Restore
[Fadahunsi and Maheswaran 2019]		Threshold		Replication
[Fahs and Pierre 2019]				Replication
[Fiandrino et al. 2019]	Encryption			
[Filiposka et al. 2019]				Migration
[Fizza et al. 2018]		Threshold	Reservation	
[Giang et al. 2020]		Threshold		
[Guerrero et al. 2019]				Migration
[He et al. 2018]			Incentives	
[Huang et al. 2019]				Replication
[Huang and Xu 2016]			Reservation	
[Josilo and Dan 2019]			Incentives	
[Kayal and Liebeherr 2019]		Threshold		
[Kim et al. 2019]		Threshold		
[Kim and Chung 2018]			Compensation	
[Lee et al. 2019]		Threshold		
[Li et al. 2019b]			Incentives	
[Li et al. 2018b]		Threshold		
[Li et al. 2019a]		Implication		
[Luo et al. 2019]	Authentication	Threshold		
[Mahmud et al. 2018c]				Migration
[Mahmud et al. 2020]			Compensation	
[Melnik et al. 2019]				Backup-Restore
[Mohamed et al. 2019]				Backup-Restore
[Mouradian et al. 2019]		Threshold		
[Nair and Somasundaram 2019]		Implication		Migration
[Noura et al. 2019]	Encryption			Backup-Restore
[Oma et al. 2019]				Backup-Restore, Replication

(Continued)

Table 5. Continued

Works	Application Maintenance			
	Security Considerations	Performance Monitor	Monetary Support	Resiliency Strategy
[Ozeer et al. 2018]		Threshold		Backup-Restore
[Prabavathy et al. 2019]		Implication, Threshold		
[Puliafito et al. 2018]				Migration
[Rahbari and Nickray 2019]	Integrity, Authentication			
[Saurez et al. 2016]	Integrity			Migration
[Su et al. 2018]	Encryption			Replication
[Tuli et al. 2019]	Integrity			
[Tychalas and Karatza 2020]		Threshold		
[Verba et al. 2019]				Migration
[Vinueza Naranjo et al. 2018]	Authentication			
[Wang et al. 2019a]		Implication		Migration
[Wang et al. 2019]		Threshold		Replication
[Wang et al. 2018]	Authentication			
[Wazid et al. 2019]	Authentication			
[Yao et al. 2019]	Authentication			Migration
[Yu et al. 2017]			Compensation	
[Zeng et al. 2018]				Replication
[Zeng et al. 2018]			Incentives	
[Battula et al. 2019]			Compensation	
[Zhao et al. 2019]				Migration
[Zhou et al. 2019]			Incentives	
[Zhu et al. 2019]				Migration

the IABs. Additionally, the CPSs can host some components of the requested applications for data pre-processing.

6.2 Fog Gateway Level

At the Fog gateway level, an IAB consists of the CPS manager, Application Placement Engine (APE) and Workload Scheduler. The CPS manager contains the meta-data of multiple versions of an application that have different programming models, functional layouts and interaction methods. The CPS manager also interacts with the APE to get the state of resources such as their orientation and type within the Fog infrastructure and Cloud level. Based on the accumulated information from different levels, the CPS manager determines the most suitable architecture of the applications for placement. Later, the APE estimates the resources of executing the application, identifies the placement metrics as per the application QoS requirements of the CPSs and sets the mapping technique accordingly. To place the applications physically over the resources, the APE communicates with the Fog Resource Manager (FRM) and Cloud Resource Manager (CRM) of the Fog infrastructure and Cloud level. After placement, the Workload Scheduler finds out the feasible placement strategy to dispatch the inputs to the applications based on the dynamics of the Fog environment.

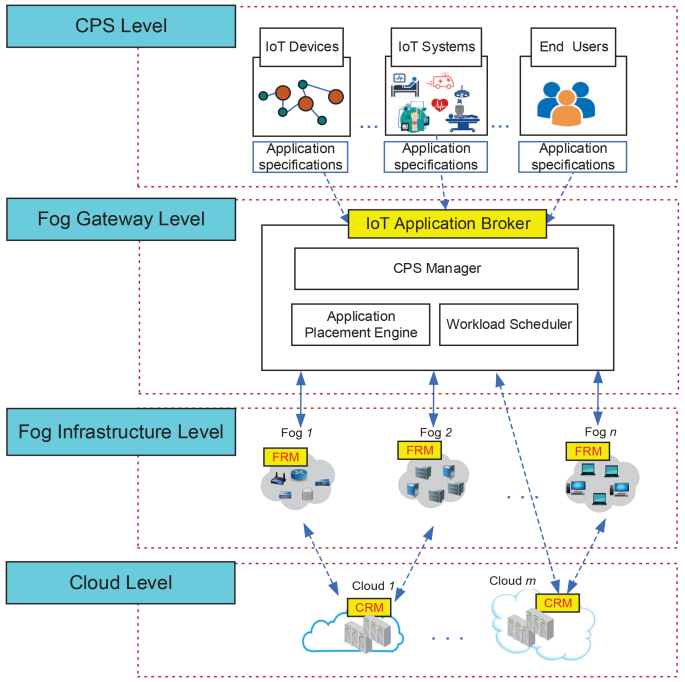

Fig. 7. A perspective model for application management in Fog.

6.3 Fog Infrastructure and Cloud Level

The FRMs and CRMs of the Fog infrastructure and Cloud datacentre store the application executables and are responsible for allocating resources for application execution. They also monitor the status and performance of the resources and conduct application maintenance operations including service backup and replication. Additionally, they deal with the uncertain node failures, resource outage and security attacks to ensure reliability during application execution. Based on their implications, the CPSs and IABs tune the specifications of the application architecture and modify the placement approaches.

Nevertheless, this framework only provides an abstract view of distributing application management operations in different infrastructure levels within the Fog computing environments. This framework can also contribute to develop new policies for runtime service orchestration, multi-level resource provisioning, application execution migration and Fog standardization.

7 FUTURE RESEARCH DIRECTIONS

In this section, we discuss several future research directions that can guide the respective community to leverage the existing solutions and make further progress in the field of Fog computing. These directions are listed next.

Trade-off between energy and accuracy. A complex relation exists among the accuracy-level applications, the sensing rate of IoT devices and the energy consumption of Fog nodes [Tuli et al. 2019]. Based on such relation, a policy to dynamically tune the accuracy level and the sensing frequency of the IoT devices can be developed for meeting the energy constraints of the Fog nodes, especially when the renewable power sources are used.

Artificial intelligence-based application management. Currently, artificial intelligence is receiving significant attention due its ability to solve complex problems. The training data required to build an artificial intelligent system is very easy to accumulate in Fog [Li et al. 2019b]. Artificial intelligence–based application management can help predict the future resource requirements, context variation and nodal failures more precisely, and manage the applications accordingly.

Pricing and detailed estimation of Fog resources. The Cloud-based pricing models for subscription-oriented services cannot be directly applied to Fog computing due to the localized demand and distributed deployment of the IoT-enabled systems. For the same reasons, resource over-provisioning can also occur in Fog computing environments [Mahmud et al. 2020]. Therefore, detailed estimation of resources in Fog computing is needed that can consider the number of IoT devices within the CPS and their future service requirements simultaneously. Such research will also help develop an efficient business model for Fog computing environments.

Trusted service orchestration in Fog. The Fog infrastructure can be private or public. The publicly available Fog infrastructure is highly exposed to security threats. However, service of privately owned Fog infrastructure is subjected to lack of transparency [Pallewatta et al. 2019]. In this case, a trusted service orchestration policy is required to ensure collaboration and reliability between different types of Fog computing infrastructure.

Fog node consolidation and scaling. Fog nodes are resource constrained. Inclusion of more Fog nodes can alleviate this limitation. However, it increases deployment cost, communication interference and energy consumption at the edge network [Afrin et al. 2019]. In this case, dynamic consolidation and scaling of Fog nodes as per the computational demand can be helpful.

Application-specific management. Fog computing is developed to execute various complex IoT applications from different domains including smart healthcare, city, agriculture and industry [Mohamed et al. 2019]. These IoT applications have specific requirements and need specialized support. Application-specific management polices can be helpful in dealing with them in Fog.

Task sharing and re-usability. Applications can share a particular task among themselves to optimize the computational load on Fog nodes [Varshney and Simmhan 2017]. In addition, the task executables of recently terminated applications can be also re-used for other applications. To perform such operations, shared caching techniques and policies must be developed in the context of Fog computing.

8 SUMMARY AND CONCLUSION

Fog computing is gradually turning into an integral component of smart systems because of its widespread features for supporting IoT-driven use cases. To exploit the benefits of Fog computing, the efficient management of applications over the Fog nodes is very important. In both academia and industry, numerous initiatives have already been taken to standardize the Fog computing concept for managing the IoT applications. In this work, we reviewed the existing application management strategies in Fog computing from the perspectives of application architecture, placement and maintenance. We proposed separate taxonomies for each of the aspects of application management and discussed their associated research gaps. We also highlighted a perspective model for managing applications in Fog environments and mentioned several research directions for further improvement of Fog computing.

REFERENCES

M. Aazam and E. Huh. 2015. Fog computing micro datacenter based dynamic resource estimation and pricing model for IoT. In *Proceedings of the 2015 IEEE 29th International Conference on Advanced Information Networking and Applications.* 687–694.

Mohammad Aazam, Sherali 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.

Sadam Hussain Abbasi, Nadeem Javaid, Muhammad Hassaan Ashraf, Mubashar Mehmood, Maria Naeem, and Mubariz Rehman. 2019. Load stabilizing in fog computing environment using load balancing algorithm. In *Advances on Broadband and Wireless Computing, Communication and Applications*. Springer International Publishing, Cham, Switzerland, 737–750.

Eman AbdElhalim, Marwa Obayya, and Sherif Kishk. 2019. Distributed fog-to-cloud computing system: A minority game approach. *Concurrency and Computation: Practice and Experience* 31, 15 (2019), e5162.

Mainak Adhikari and Hemant Gianey. 2019. Energy efficient offloading strategy in fog-cloud environment for IoT applications. *Internet of Things* 6 (2019), 100053.

M. Adhikari, M. Mukherjee, and S. N. Srirama. 2019. DPTO: A deadline and priority-aware task offloading in fog computing framework leveraging multi-level feedback queueing. *IEEE Internet of Things Journal*. Accepted. DOI: https://doi.org/10.1109/JIOT.2019.2946426

Mahbuba Afrin, Jiong Jin, and Ashfaqur Rahman. 2018. Energy-delay co-optimization of resource allocation for robotic services in cloudlet infrastructure. In *Proceedings of the International Conference on Service-Oriented Computing*. 295–303.

Mahbuba Afrin, Jiong Jin, Ashfaqur Rahman, Yu-Chu Tian, and Ambarish Kulkarni. 2019. Multi-objective resource allocation for edge cloud based robotic workflow in smart factory. *Future Generation Computer Systems* 97 (2019), 119–130.

M. Afrin, M. R. Mahmud, and M. A. Razzaque. 2015. Real time detection of speed breakers and warning system for on-road drivers. In *Proceedings of the 2015 IEEE International WIE Conference on Electrical and Computer Engineering (WIECONECE'15)*. 495–498.

Mahbuba Afrin, Md Razzaque, Iffat Anjum, Mohammad Mehedi Hassan, and Atif Alamri. 2017. Tradeoff between user quality-of-experience and service provider profit in 5G cloud radio access network. *Sustainability* 9, 11 (2017), 2127.

Surin Ahn, Maria Gorlatova, Parinaz Naghizadeh, Mung Chiang, and Prateek Mittal. 2018. Adaptive fog-based output security for augmented reality. In *Proceedings of the 2018 Morning Workshop on Virtual Reality and Augmented Reality Network (VR/AR Network'18)*. ACM, New York, NY, 1–6.

M. Al-Khafajiy, T. Baker, H. Al-Libawy, Z. Maamar, M. Aloqaily, and Y. Jararweh. 2019. Improving fog computing performance via fog-2-fog collaboration. *Future Generation Computer Systems* 100 (2019), 260–280.

M. Ali, N. Riaz, M. I. Ashraf, S. Qaisar, and M. Naeem. 2018. Joint cloudlet selection and latency minimization in fog networks. *IEEE Transactions on Industrial Informatics* 14, 9 (Sept. 2018), 4055–4063.

Adam A. Alli and Muhammad Mahbub Alam. 2019. SecOFF-FCIoT: Machine learning based secure offloading in fog-cloud of things for smart city applications. *Internet of Things* 7 (2019), 100070.

A. Alnoman and A. Anpalagan. 2018. A dynamic priority service provision scheme for delay-sensitive applications in fog computing. In *Proceedings of the 2018 29th Biennial Symposium on Communications (BSC'18)*. IEEE, Los Alamitos, CA, 1–5.

A. Alrawais, A. Alhothaily, C. Hu, X. Xing, and X. Cheng. 2017. An attribute-based encryption scheme to secure fog communications. *IEEE Access* 5 (2017), 9131–9138.

Cosimo Anglano, Massimo Canonico, Paolo Castagno, Marco Guazzone, and Matteo Sereno. 2019b. Profit-aware coalition formation in fog computing providers: A game-theoretic approach. *Concurrency and Computation: Practice and Experience*. In Press.

Cosimo Anglano, Massimo Canonico, and Marco Guazzone. 2019a. Online user-driven task scheduling for FemtoClouds. In *Proceedings of the 2019 4th International Conference on Fog and Mobile Edge Computing (FMEC'19)*. 5–12.

Arman Anzanpour, Humayun Rashid, Amir M. Rahmani, Axel Jantsch, Nikil Dutt, and Pasi Liljeberg. 2019. Energy-efficient and reliable wearable Internet-of-Things through fog-assisted dynamic goal management. *Procedia Computer Science* 151 (2019), 493–500.

H. R. Arkian, A. Diyanat, and A. 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.

Deeksha Arya and Mayank Dave. 2017. Priority based service broker policy for fog computing environment. In *Advanced Informatics for Computing Research*. Springer Singapore, Singapore, 84–93.

N. Auluck, A. Azim, and K. Fizza. 2019. Improving the schedulability of real-time tasks using fog computing. *IEEE Transactions on Services Computing*. Early Access. September 27, 2019. DOI: https://doi.org/10.1109/TSC.2019.2944360

M. Avgeris, D. Dechouniotis, N. Athanasopoulos, and S. Papavassiliou. 2019. Adaptive resource allocation for computation offloading: A control-theoretic approach. *ACM Transactions on Internet Technology* 19, 2 (April 2019), Article 23, 20 pages.

E. Baccarelli, P. G. V. Naranjo, M. Scarpiniti, M. Shojafar, and J. H. Abawajy. 2017. Fog of everything: Energy-efficient networked computing architectures, research challenges, and a case study. *IEEE Access* 5 (2017), 9882–9910.

R. K. Barik, A. C. Dubey, A. Tripathi, T. Pratik, S. Sasane, R. K. Lenka, H. Dubey, K. Mankodiya, and V. Kumar. 2018. Mist data: Leveraging mist computing for secure and scalable architecture for smart and connected health. *Procedia Computer Science* 125 (2018), 647–653.

Sudheer Kumar Battula, Saurabh Garg, Ranesh Kumar Naha, Parimala Thulasiraman, and Ruppa Thulasiram. 2019. A micro-level compensation-based cost model for resource allocation in a fog environment. *Sensors* 19, 13 (2019), 2954.

Ranjit Kumar Behera, K. Hemant Kumar Reddy, and Diptendu Sinha Roy. 2020. A novel context migration model for fog-enabled cross-vertical IoT applications. In *Proceedings of the International Conference on Innovative Computing and Communications*. 287–295.

Paolo Bellavista, Javier Berrocal, Antonio Corradi, Sajal K. Das, Luca Foschini, and Alessandro Zanni. 2019. A survey on fog computing for the Internet of Things. *Pervasive and Mobile Computing* 52 (2019), 71–99.

Paolo Bellavista, Antonio Corradi, Luca Foschini, and Domenico Scotece. 2019. Differentiated service/data migration for edge services leveraging container characteristics. *IEEE Access* 7 (2019), 139746–139758.

L. Belli, S. Cirani, L. Davoli, A. Gorrieri, M. Mancin, M. Picone, and G. Ferrari. 2015. Design and deployment of an IoT application-oriented testbed. *Computer* 48, 9 (Sept. 2015), 32–40.

Amira Rayane Benamer, Hana Teyeb, and Nejib Ben Hadj-Alouane. 2018. Latency-aware placement heuristic in fog computing environment. In *On the Move to Meaningful Internet Systems. OTM 2018*. Lecture Notes in Computer Science, Vol. 11230. Springer, 241–257.

M. A. Benblidia, B. Brik, L. Merghem-Boulahia, and M. Esseghir. 2019. Ranking fog nodes for tasks scheduling in fog-cloud environments: A fuzzy logic approach. In *Proceedings of the 2019 15th International Wireless Communications Mobile Computing Conference (IWCMC'19)*. 1451–1457.

Munish Bhatia, Sandeep K. Sood, and Simranpreet Kaur. 2019. Quantum-based predictive fog scheduler for IoT applications. *Computers in Industry* 111 (2019), 51–67.

Huynh Thi Thanh Binh, Tran The Anh, Do Bao Son, Pham Anh Duc, and Binh Minh Nguyen. 2018. An evolutionary algorithm for solving task scheduling problem in cloud-fog computing environment. In *Proceedings of the 9th International Symposium on Information and Communication Technology (SoICT'18)*. ACM, New York, NY, 397–404.

F. Bonomi, R. Milito, J. Zhu, and S. Addepalli. 2012. Fog computing and its role in the Internet of Things. In *Proceedings of the 1st Edition of the MCC Workshop on Mobile Cloud Computing*. ACM, New York, NY, 13–16.

A. Brogi and S. Forti. 2017. QoS-aware deployment of IoT applications through the fog. *IEEE Internet of Things Journal* 4, 5 (Oct. 2017), 1185–1192.

Rajkumar Buyya, Chee Shin Yeo, Srikumar Venugopal, James Broberg, and Ivona Brandic. 2009. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems* 25, 6 (2009), 599–616.

Danilo Charântola, Alexandre C. Mestre, Rafael Zane, and Luiz F. Bittencourt. 2019. Component-based scheduling for fog computing. In *Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC'19 Companion)*. ACM, New York, NY, 3–8.

Min Chen, Wei Li, Giancarlo Fortino, Yixue Hao, Long Hu, and Iztok Humar. 2019. A dynamic service migration mechanism in edge cognitive computing. *ACM Transactions on Internet Technology* 19, 2 (April 2019), Article 30, 15 pages.

B. Cheng, G. Solmaz, F. Cirillo, E. Kovacs, K. Terasawa, and A. Kitazawa. 2018. FogFlow: Easy programming of IoT services over cloud and edges for smart cities. *IEEE Internet of Things Journal* 5, 2 (April 2018), 696–707.

Francesco Chiti, Romano Fantacci, and Benedetta Picano. 2019. A matching game for tasks offloading in integrated edge-fog computing systems. *Transactions on Emerging Telecommunications Technologies* 31, 2 (2019), e3718.

Tejaswini Choudhari, Melody Moh, and Teng-Sheng Moh. 2018. Prioritized task scheduling in fog computing. In *Proceedings of the ACMSE 2018 Conference (ACMSE'18)*. ACM, New York, NY, Article 22, 8 pages.

Abdullahi Chowdhury, Gour Karmakar, and Joarder Kamruzzaman. 2019. The co-evolution of cloud and IoT applications: Recent and future trends. In *Handbook of Research on the IoT, Cloud Computing, and Wireless Network Optimization*. IGI Global, 213–234.

Federico Concone, Giuseppe Lo Re, and Marco Morana. 2019. A fog-based application for human activity recognition using personal smart devices. *ACM Transactions on Internet Technology* 19, 2 (March 2019), Article 20, 20 pages.

G. Cristescu, R. Dobrescu, O. Chenaru, and G. Florea. 2019. DEW: A new edge computing component for distributed dynamic networks. In *Proceedings of the 2019 22nd International Conference on Control Systems and Computer Science (CSCS'19)*. 547–551.

L. Dang, M. Dong, K. Ota, J. Wu, J. Li, and G. Li. 2018. Resource-efficient secure data sharing for information centric e-health system using fog computing. In *Proceedings of the 2018 IEEE International Conference on Communications (ICC'18)*. 1–6.

A. V. Dastjerdi, H. Gupta, R. N. Calheiros, S. K. Ghosh, and R. Buyya. 2016. Fog computing: Principles, architectures, and applications. In *Internet of Things: Principles and Paradigms*. Morgan Kaufmann, 61–75. DOI: https://doi.org/10.1016/B978-0-12-805395-9.00004-6

Jean Lucas de Souza Toniolli and Brigitte Jaumard. 2019. Resource allocation for multiple workflows in cloud-fog computing systems. In *Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC'19 Companion)*. ACM, New York, NY, 77–84.

S. Dehnavi, H. R. Faragardi, M. Kargahi, and T. Fahringer. 2019. A reliability-aware resource provisioning scheme for real-time industrial applications in a Fog-integrated smart factory. *Microprocessors and Microsystems* 70 (2019), 1–14.

R. Deng, R. Lu, C. Lai, T. H. Luan, and H. Liang. 2016. Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption. *IEEE Internet of Things Journal* 3, 6 (Dec. 2016), 1171–1181.

Ruimiao Ding, Xuejun Li, Xiao Liu, and Jia Xu. 2019. A cost-effective time-constrained multi-workflow scheduling strategy in fog computing. In *Service-Oriented Computing—ICSOC 2018 Workshops*. Lecture Notes in Computer Science, Vol. 11434. Springer, 194–207.

T. Djemai, P. Stolf, T. Monteil, and J. Pierson. 2019. A discrete particle swarm optimization approach for energy-efficient IoT services placement over fog infrastructures. In *Proceedings of the 2019 18th International Symposium on Parallel and Distributed Computing (ISPDC'19)*. 32–40.

J. Du, L. Zhao, J. Feng, and X. Chu. 2018. Computation offloading and resource allocation in mixed fog/cloud computing systems with min-max fairness guarantee. *IEEE Transactions on Communications* 66, 4 (April 2018), 1594–1608.

Mohammed S. Elbamby, Mehdi Bennis, Walid Saad, Matti Latva-Aho, and Choong Seon Hong. 2018. Proactive edge computing in fog networks with latency and reliability guarantees. *EURASIP Journal on Wireless Communications and Networking* 2018, 1 (Aug. 2018), 209.

Olamilekan Fadahunsi and Muthucumaru Maheswaran. 2019. Locality sensitive request distribution for fog and cloud servers. *Service Oriented Computing and Applications* 13, 2 (June 2019), 127–140.

A. J. Fahs and G. Pierre. 2019. Proximity-aware traffic routing in distributed fog computing platforms. In *Proceedings of the 2019 19th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing (CCGRID'19)*. 478–487.

J. Fan, X. Wei, T. Wang, T. Lan, and S. Subramaniam. 2017. Deadline-aware task scheduling in a tiered IoT infrastructure. In *Proceedings of the 2017 IEEE Global Communications Conference (GLOBECOM'17)*. 1–7.

Weidong Fang, Wuxiong Zhang, Wei Chen, Yang Liu, and Chaogang Tang. 2019. TMSRS: Trust management-based secure routing scheme in industrial wireless sensor network with fog computing. *Wireless Networks* 26 (Sept. 2019), 3169–3182.

Peter Farhat, Hani Sami, and Azzam Mourad. 2019. Reinforcement R-learning model for time scheduling of on-demand fog placement. *Journal of Supercomputing* 76 (Oct. 2019), 388–410.

C. Fiandrino, N. Allio, D. Kliazovich, P. Giaccone, and P. Bouvry. 2019. Profiling performance of application partitioning for wearable devices in mobile cloud and fog computing. *IEEE Access* 7 (2019), 12156–12166.

Sonja Filiposka, Anastas Mishev, and Katja Gilly. 2019. Mobile-aware dynamic resource management for edge computing. *Transactions on Emerging Telecommunications Technologies* 30, 6 (2019), e3626.

K. Fizza, N. Auluck, O. Rana, and L. Bittencourt. 2018. PASHE: Privacy aware scheduling in a heterogeneous fog environment. In *Proceedings of the 2018 IEEE 6th International Conference on Future Internet of Things and Cloud (FiCloud'18)*. 333–340.

Ahmed A. A. Gad-Elrab and Amin Y. Noaman. 2020. A two-tier bipartite graph task allocation approach based on fuzzy clustering in cloud–fog environment. *Future Generation Computer Systems* 103 (2020), 79–90.

Pegah Gazori, Dadmehr Rahbari, and Mohsen Nickray. 2019. Saving time and cost on the scheduling of fog-based IoT applications using deep reinforcement learning approach. *Future Generation Computer Systems* 110 (2019), 1098–1115. DOI: https://doi.org/10.1016/j.future.2019.09.060

Mostafa Ghobaei-Arani, Alireza Souri, and Ali A Rahmanian. 2019. Resource management approaches in fog computing: A comprehensive review. *Journal of Grid Computing* 18 (2019), 1–42.

S. Ghosh, A. Mukherjee, S. K. Ghosh, and R. Buyya. 2019. Mobi-IoST: Mobility-aware cloud-fog-edge-iot collaborative framework for time-critical applications. *IEEE Transactions on Network Science and Engineering*. Early Access. September 16, 2019. DOI: https://doi.org/10.1109/TNSE.2019.2941754

Nam Ky Giang, Rodger Lea, and Victor C. M. Leung. 2020. Developing applications in large scale, dynamic fog computing: A case study. *Software: Practice and Experience* 50, 5 (2020), 519–532. DOI: https://doi.org/10.1002/spe.2695

Mohammad Goudarzi, Marimuthu Palaniswami, and Rajkumar Buyya. 2019. A fog-driven dynamic resource allocation technique in ultra dense femtocell networks. *Journal of Network and Computer Applications* 145, 1 (2019), 102407.

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.

Carlos Guerrero, Isaac Lera, and Carlos Juiz. 2019. A lightweight decentralized service placement policy for performance optimization in fog computing. *Journal of Ambient Intelligence and Humanized Computing* 10, 6 (June 2019), 2435–2452.

M. A. Hassan, M. Xiao, Q. Wei, and S. Chen. 2015. Help your mobile applications with fog computing. In *Proceedings of the 2015 12th Annual IEEE International Conference on Sensing, Communication, and Networking Workshops (SECON Workshops'15)*. 1–6.

J. He, J. Wei, K. Chen, Z. Tang, Y. Zhou, and Y. Zhang. 2018. Multitier fog computing with large-scale IoT data analytics for smart cities. *IEEE Internet of Things Journal* 5, 2 (April 2018), 677–686.

Xiang He, Zhiying Tu, Xiaofei Xu, and Zhongjie Wang. 2019. Re-deploying microservices in edge and cloud environment for the optimization of user-perceived service quality. In *Service-Oriented Computing*, S. Yangui, I. B. Rodriguez, K. Drira, and Z. Tari (Eds.). Springer International Publishing, Cham, Switzerland, 555–560.

Cheol-Ho Hong and Blesson Varghese. 2018. Resource management in fog/edge computing: A survey. arXiv:1810.00305. http://arxiv.org/abs/1810.00305

Pengfei Hu, Sahraoui Dhelim, Huansheng Ning, and Tie Qiu. 2017. Survey on fog computing: Architecture, key technologies, applications and open issues. *Journal of Network and Computer Applications* 98 (2017), 27–42.

C. Huang and K. Xu. 2016. Reliable realtime streaming in vehicular cloud-fog computing networks. In *Proceedings of the 2016 IEEE/CIC International Conference on Communications in China (ICCC'16)*. 1–6.

Tiansheng Huang, Weiwei Lin, Yin Li, LiGang He, and ShaoLiang Peng. 2019. A latency-aware multiple data replicas placement strategy for fog computing. *Journal of Signal Processing Systems* 91, 10 (Oct. 2019), 1191–1204.

S. Imai, C. A. Varela, and S. Patterson. 2018. A performance study of geo-distributed IoT data aggregation for fog computing. In *Proceedings of the 2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion*. 278–283.

IoT for All. 2018. The Big Three Make a Play for the Fog. Retrieved April 8, 2020 from https://www.iotforall.com/big-three-make-play-fog/.

Bushra Jamil, Mohammad Shojafar, Israr Ahmed, Atta Ullah, Kashif Munir, and Humaira Ijaz. 2019. A job scheduling algorithm for delay and performance optimization in fog computing. *Concurrency and Computation: Practice and Experience* 32, 7 (2019), e5581. DOI: https://doi.org/10.1002/cpe.5581

T. Jeong, J. Chung, J. W. Hong, and S. Ha. 2017. Towards a distributed computing framework for Fog. In *Proceedings of the 2017 IEEE Fog World Congress (FWC'17)*. 1–6.

Y. Jiang, Y. Chen, S. Yang, and C. Wu. 2019. Energy-efficient task offloading for time-sensitive applications in fog computing. *IEEE Systems Journal* 13, 3 (Sept. 2019), 2930–2941.

S. Josilo and G. Dan. 2019. Decentralized algorithm for randomized task allocation in fog computing systems. *IEEE/ACM Transactions on Networking* 27, 1 (Feb. 2019), 85–97.

Muhammad Babar Kamal, Nadeem Javaid, Syed Aon Ali Naqvi, Hanan Butt, Talha Saif, and Muhammad Daud Kamal. 2019. Heuristic min-conflicts optimizing technique for load balancing on fog computing. In *Advances in Intelligent Networking and Collaborative Systems*, F. Xhafa, L. Barolli, and M.Greguš (Eds.). Springer International Publishing, Cham, Switzerland, 207–219.

A. Karamoozian, A. Hafid, and E. M. Aboulhamid. 2019. On the fog-cloud cooperation: How fog computing can address latency concerns of IoT applications. In *Proceedings of the 4th International Conference on Fog and Mobile Edge Computing*. 166–172.

Firat Karatas and Ibrahim Korpeoglu. 2019. Fog-based data distribution service (F-DAD) for Internet of Things (IoT) applications. *Future Generation Computer Systems* 93 (2019), 156–169.

P. Kayal and J. Liebeherr. 2019. Autonomic service placement in fog computing. In *Proceedings of the 2019 IEEE 20th International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM'19)*. 1–9.

Bongjun Kim, Seonyeong Heo, Gyeongmin Lee, Seungbin Song, Jong Kim, and Hanjun Kim. 2019. Spinal code: Automatic code extraction for near-user computation in fogs. In *Proceedings of the 28th International Conference on Compiler Construction (CC'19)*. ACM, New York, NY, 87–98.

Won-Suk Kim and Sang-Hwa Chung. 2018. User incentive model and its optimization scheme in user-participatory fog computing environment. *Computer Networks* 145 (2018), 76–88.

Guenter I. Klas. 2015. Fog computing and mobile edge cloud gain momentum Open Fog Consortium, ETSI MEC and cloudlets. Retrieved 8 April, 2020 from https://yucianga.info/?p=938.

Frank Alexander Kraemer, Anders Eivind Braten, Nattachart Tamkittikhun, and David Palma. 2017. Fog computing in healthcare—A review and discussion. *IEEE Access* 5 (2017), 9206–9222.

G. Lee, W. Saad, and M. Bennis. 2019. An online optimization framework for distributed fog network formation with minimal latency. *IEEE Transactions on Wireless Communications* 18, 4 (April 2019), 2244–2258.

I. Lera, C. Guerrero, and C. Juiz. 2019. Availability-aware service placement policy in fog computing based on graph partitions. *IEEE Internet of Things Journal* 6, 2 (April 2019), 3641–3651.

Chao Li, Yushu Xue, Jing Wang, Weigong Zhang, and Tao Li. 2018a. Edge-oriented computing paradigms: A survey on architecture design and system management. *ACM Computing Surveys* 51, 2 (2018), 39.

Changlong Li, Hang Zhuang, Qingfeng Wang, and Xuehai Zhou. 2018b. SSLB: Self-similarity-based load balancing for large-scale fog computing. *Arabian Journal for Science and Engineering* 43, 12 (Dec. 2018), 7487–7498.

Guangshun Li, Jiahe Yan, Lu Chen, Junhua Wu, Qingyan Lin, and Ying Zhang. 2019c. Energy consumption optimization with a delay threshold in cloud-fog cooperation computing. *IEEE Access* 7 (2019), 159688–159697.

He Li, Kaoru Ota, and Mianxiong Dong. 2019b. Deep reinforcement scheduling for mobile crowdsensing in fog computing. *ACM Transactions on Internet Technology* 19, 2 (April 2019), Article 21, 18 pages.

Lei Li, Quansheng Guan, Lianwen Jin, and Mian Guo. 2019a. Resource allocation and task offloading for heterogeneous real-time tasks with uncertain duration time in a fog queueing system. *IEEE Access* 7 (2019), 9912–9925.

Songze Li, Mohammad Ali Maddah-Ali, and A. Salman Avestimehr. 2017. Coding for distributed fog computing. *IEEE Communications Magazine* 55, 4 (2017), 34–40.

Liqing Liu, Zheng Chang, Xijuan Guo, and T. Ristaniemi. 2017. Multi-objective optimization for computation offloading in mobile-edge computing. In *Proceedings of the 2017 IEEE Symposium on Computers and Communications (ISCC'17)*. 832–837.

Z. Liu, X. Yang, Y. Yang, K. Wang, and G. Mao. 2019. DATS: Dispersive stable task scheduling in heterogeneous fog networks. *IEEE Internet of Things Journal* 6, 2 (April 2019), 3423–3436.

J. Luo, L. Yin, J. Hu, C. Wang, X. Liu, X. Fan, and H. Luo. 2019. Container-based fog computing architecture and energy-balancing scheduling algorithm for energy IoT. *Future Generation Computer Systems* 97 (2019), 50–60.

H. Madsen, B. Burtschy, G. Albeanu, and F. Popentiu-Vladicescu. 2013. Reliability in the utility computing era: Towards reliable Fog computing. In *Proceedings of the 2013 20th International Conference on Systems, Signals, and Image Processing (IWSSIP'13)*. 43–46.

Mukhtar M. E. Mahmoud, Joel J. P. C. Rodrigues, Kashif Saleem, Jalal Al-Muhtadi, Neeraj Kumar, and Valery Korotaev. 2018. Towards energy-aware fog-enabled cloud of things for healthcare. *Computers & Electrical Engineering* 67 (2018), 58–69.

Md. Redowan Mahmud, Mahbuba Afrin, Md. Abdur Razzaque, Mohammad Mehedi Hassan, Abdulhameed Alelaiwi, and Majed Alrubaian. 2016. Maximizing quality of experience through context-aware mobile application scheduling in cloudlet infrastructure. *Software: Practice and Experience* 46, 11 (2016), 1525–1545.

Redowan Mahmud, Fernando Luiz Koch, and Rajkumar Buyya. 2018a. Cloud-fog interoperability in IoT-enabled healthcare solutions. In *Proceedings of the 19th International Conference on Distributed Computing and Networking (ICDCN'18)*. ACM, New York, NY, Article 32, 10 pages.

Redowan Mahmud, Ramamohanarao Kotagiri, and Rajkumar Buyya. 2018b. Fog computing: A taxonomy, survey and future directions. In *Internet of Everything*. Springer, 103–130.

Redowan Mahmud, Kotagiri Ramamohanarao, and Rajkumar Buyya. 2018c. Latency-aware application module management for fog computing environments. *ACM Transactions on Internet Technology* 19, 1 (Nov. 2018), Article 9, 21 pages.

Redowan Mahmud, Kotagiri Ramamohanarao, and Rajkumar Buyya. 2019a. Edge affinity-based management of applications in fog computing environments. In *Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing (UCC'19)*. ACM, New York, NY, 1–10.

Redowan Mahmud, Satish Narayana Srirama, Kotagiri Ramamohanarao, and Rajkumar Buyya. 2019b. Quality of experience (QoE)-aware placement of applications in Fog computing environments. *Journal of Parallel and Distributed Computing* 132 (2019), 190–203.

Redowan Mahmud, Satish Narayana Srirama, Kotagiri Ramamohanarao, and Rajkumar Buyya. 2020. Profit-aware application placement for integrated Fog–Cloud computing environments. *Journal of Parallel and Distributed Computing* 135 (2020), 177–190.

R. Mahmud, A. N. Toosi, K. Rao, and R. Buyya. 2019. Context-aware placement of Industry 4.0 applications in fog computing environments. *IEEE Transactions on Industrial Informatics*. Early Access. November 8, 2019.

Mirjana Maksimović. 2018. Implementation of fog computing in IoT-based healthcare system. *JITA—Journal of Information Technology and Applications* 14, 2 (2018), 100–107.

B. Martinez, M. Monton, I. Vilajosana, and J. D. Prades. 2015. The power of models: Modeling power consumption for IoT devices. *IEEE Sensors Journal* 15, 10 (Oct. 2015), 5777–5789.

Jakob Mass, Chii Chang, and Satish Narayana Srirama. 2018. Context-aware edge process management for mobile thing-to-fog environment. In *Proceedings of the 12th European Conference on Software Architecture: Companion Proceedings (ECSA'18)*. ACM, New York, NY, Article 44, 7 pages.

Houssemeddine Mazouzi, Nadjib Achir, and Khaled Boussetta. 2019. DM2-ECOP: An efficient computation offloading policy for multi-user multi-cloudlet mobile edge computing environment. *ACM Transactions on Internet Technology* 19, 2 (April 2019), Article 24, 24 pages.

S. Meixner, D. Schall, F. Li, V. Karagiannis, S. Schulte, and K. Plakidas. 2019. Automatic application placement and adaptation in cloud-edge environments. In *Proceedings of the 2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA'19)*. 1001–1008.

Eduard Melnik, Anna Klimenko, and Vladislav Klimenko. 2019. A recovery technique for the fog-computing-based information and control systems. In *Intelligent Systems in Cybernetics and Automation Control Theory*, R. Silhavy, P. Silhavy, and Z. Prokopova (Eds.). Springer International Publishing, Cham, Switzerland, 216–227.

Moumita Mishra, Sayan Kumar Roy, Anwesha Mukherjee, Debashis De, Soumya K. Ghosh, and Rajkumar Buyya. 2019. An energy-aware multi-sensor geo-fog paradigm for mission critical applications. *Journal of Ambient Intelligence and Humanized Computing*. Early Access. September 12, 2019.

S. K. Mishra, D. Puthal, J. J. P. C. Rodrigues, B. Sahoo, and E. Dutkiewicz. 2018. Sustainable service allocation using a metaheuristic technique in a fog server for industrial applications. *IEEE Transactions on Industrial Informatics* 14, 10 (Oct. 2018), 4497–4506.

N. Mohamed, J. Al-Jaroodi, and I. Jawhar. 2019. Towards fault tolerant fog computing for IoT-based smart city applications. In *Proceedings of the 2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC'19)*. 0752–0757.

C. Mouradian, S. Kianpisheh, M. Abu-Lebdeh, F. Ebrahimnezhad, N. T. Jahromi, and R. H. Glitho. 2019. Application component placement in NFV-based hybrid cloud/fog systems with mobile fog nodes. *IEEE Journal on Selected Areas in Communications* 37, 5 (May 2019), 1130–1143.

Carla Mouradian, Diala Naboulsi, Sami Yangui, Roch H. Glitho, Monique J. Morrow, and Paul A. Polakos. 2017. A comprehensive survey on fog computing: State-of-the-art and research challenges. *IEEE Communications Surveys & Tutorials* 20, 1 (2017), 416–464.

M. Mtshali, H. Kobo, S. Dlamini, M. Adigun, and P. Mudali. 2019. Multi-objective optimization approach for task scheduling in fog computing. In *Proceedings of the International Conference on Advances in Big Data, Computing, and Data Communication Systems*. 1–6.

Mithun Mukherjee, Lei Shu, and Di Wang. 2018. Survey of fog computing: Fundamental, network applications, and research challenges. *IEEE Communications Surveys & Tutorials* 20, 3 (2018), 1826–1857.

Mohammed Islam NAAS, Laurent Lemarchand, Jalil Boukhobza, and Philippe Raipin. 2018. A graph partitioning-based heuristic for runtime IoT data placement strategies in a fog infrastructure. In *Proceedings of the 33rd Annual ACM Symposium on Applied Computing (SAC'18)*. ACM, New York, NY, 767–774.

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.

Biji Nair and Mary Saira Bhanu Somasundaram. 2019. Overload prediction and avoidance for maintaining optimal working condition in a fog node. *Computers & Electrical Engineering* 77 (2019), 147–162.

Y. Nan, W. Li, W. Bao, F. C. Delicato, P. F. Pires, Y. Dou, and A. Y. Zomaya. 2017. Adaptive energy-aware computation offloading for cloud of things systems. *IEEE Access* 5 (2017), 23947–23957.

Mansoor Nasir, Khan Muhammad, Jaime Lloret, Arun Kumar Sangaiah, and Muhammad Sajjad. 2019. Fog computing enabled cost-effective distributed summarization of surveillance videos for smart cities. *Journal on Parallel and Distributed Computing* 126 (2019), 161–170.

Shubha Brata Nath, Harshit Gupta, Sandip Chakraborty, and Soumya K. Ghosh. 2018. A survey of fog computing and communication: Current researches and future directions. arXiv:1804.04365. arxiv:1804.04365 http://arxiv.org/abs/1804.04365

T. Nazar, N. Javaid, M. Waheed, A. Fatima, H. Bano, and N. Ahmed. 2019. Modified shortest job first for load balancing in cloud-fog computing. In *Advances on Broadband and Wireless Computing, Communication and Applications*, L. Barolli, F.-Y. Leu, T. Enokido, and H.-C. Chen (Eds.). Springer International Publishing, Cham, Switzerland, 63–76.

Y. Niu, Y. Liu, Y. Li, Z. Zhong, B. Ai, and P. Hui. 2018. Mobility-aware caching scheduling for fog computing in mmWave band. *IEEE Access* 6 (2018), 69358–69370.

Hassan Noura, Ola Salman, Ali Chehab, and Raphael Couturier. 2019. Preserving data security in distributed fog computing. *Ad Hoc Networks* 94 (2019), 101937.

Ryuji Oma, Shigenari Nakamura, Dilawaer Duolikun, Tomoya Enokido, and Makoto Takizawa. 2019. Fault-tolerant fog computing models in the IoT. In *Advances on P2P, Parallel, Grid, Cloud and Internet Computing*, F. Xhafa, F.-Y. Leu, M. Ficco, and C.-T. Yang (Eds.). Springer International Publishing, Cham, Switzerland, 14–25.

Opeyemi Osanaiye, Shuo Chen, Zheng Yan, Rongxing Lu, Kim-Kwang Raymond Choo, and Mqhele Dlodlo. 2017. From cloud to fog computing: A review and a conceptual live VM migration framework. *IEEE Access* 5 (2017), 8284–8300.

Umar Ozeer, Xavier Etchevers, Loïc Letondeur, François-Gaël Ottogalli, Gwen Salaün, and Jean-Marc Vincent. 2018. Resilience of stateful IoT applications in a dynamic fog environment. In *Proceedings of the 15th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking, and Services*. ACM, New York, NY, 332–341.

Samodha Pallewatta, Vassilis Kostakos, and Rajkumar Buyya. 2019. Microservices-based IoT application placement within heterogeneous and resource constrained fog computing environments. In *Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing*. ACM, New York, NY, 71–81.

X. Pang, Z. Bie, and X. Lin. 2018. Access point decoding coded MapReduce for tree fog network. In *Proceedings of the 2018 International Conference on Network Infrastructure and Digital Content (IC-NIDC'18)*. 384–388.

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* 50, 3 (2017), 32.

S. Prabavathy, K. Sundarakantham, S. Mercy Shalinie, and K. Narasimha Mallikarjunan. 2019. Fog computing-based autonomic security approach to Internet of Things applications. In *Computational Intelligence: Theories, Applications and Future Directions—Volume II*, N. K. Verma and A. K. Ghosh (Eds.). Springer Singapore, Singapore, 3–14.

J. S. Preden, K. Tammemae, A. Jantsch, M. Leier, A. Riid, and E. Calis. 2015. The benefits of self-awareness and attention in fog and mist computing. *Computer* 48, 7 (July 2015), 37–45.

Carlo Puliafito, Enzo Mingozzi, Francesco Longo, Antonio Puliafito, and Omer Rana. 2019. Fog computing for the Internet of Things: A survey. *ACM Transactions on Internet Technology* 19, 2 (2019), 18.

C. Puliafito, E. Mingozzi, C. Vallati, F. Longo, and G. Merlino. 2018. Companion fog computing: Supporting things mobility through container migration at the edge. In *Proceedings of the IEEE International Conference on Smart Computing*. 97–105.

Dadmehr Rahbari and Mohsen Nickray. 2019. Task offloading in mobile fog computing by classification and regression tree. *Peer-to-Peer Networking and Applications* 13 (Feb. 2019), 104–122.

M. R. Ramli, P. T. Daely, J. Lee, and D. Kim. 2019. Bio-inspired service provisioning scheme for fog-based Industrial Internet of Things. In *Proceedings of the 24th IEEE International Conference on Emerging Technologies and Factory Automation*. 1661–1664.

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.

J. Santos, T. Wauters, B. Volckaert, and F. De Turck. 2019. Towards network-aware resource provisioning in Kubernetes for fog computing applications. In *Proceedings of the 2019 IEEE Conference on Network Softwarization (NetSoft'19)*. 351–359.

S. Sarkar, S. Chatterjee, and S. Misra. 2015. Assessment of the suitability of fog computing in the context of Internet of Things. *IEEE Transactions on Cloud Computing* PP, 99 (2015), 1.

M. Satyanarayanan. 2017. The emergence of edge computing. *Computer* 50, 1 (Jan. 2017), 30–39.

Enrique Saurez, Kirak Hong, Dave Lillethun, Umakishore Ramachandran, and Beate Ottenwälder. 2016. Incremental deployment and migration of geo-distributed situation awareness applications in the fog. In *Proceedings of the 10th ACM International Conference on Distributed and Event-Based Systems (DEBS'16)*. ACM, New York, NY, 258–269.

H. Shah-Mansouri and V. W. S. Wong. 2018. Hierarchical fog-cloud computing for IoT systems: A computation offloading game. *IEEE Internet of Things Journal* 5, 4 (Aug. 2018), 3246–3257.

Shivi Sharma and Hemraj Saini. 2019. A novel four-tier architecture for delay aware scheduling and load balancing in fog environment. *Sustainable Computing: Informatics and Systems* 24 (2019), 100355.

W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu. 2016. Edge computing: Vision and challenges. *IEEE Internet of Things Journal* 3, 5 (Oct. 2016), 637–646.

W. Shi and S. Dustdar. 2016. The promise of edge computing. *Computer* 49, 5 (May 2016), 78–81.

Syed Noorulhassan Shirazi, Antonios Gouglidis, Arsham Farshad, and David Hutchison. 2017. The extended cloud: Review and analysis of mobile edge computing and fog from a security and resilience perspective. *IEEE Journal on Selected Areas in Communications* 35, 11 (2017), 2586–2595.

Leila Shooshtarian, Dapeng Lan, and Amir Taherkordi. 2019. A clustering-based approach to efficient resource allocation in fog computing. In *Pervasive Systems, Algorithms and Networks*, C. Esposito, J. Hong, and K.-K. Raymond Choo (Eds.). Springer International Publishing, Cham, Switzerland, 207–224.

Anil Singh and Nitin Auluck. 2019. Load balancing aware scheduling algorithms for fog networks. *Software: Practice and Experience*. Early Access. June 18, 2019. DOI: https://doi.org/10.1002/spe.2722

Simar Preet Singh, Anju Sharma, and Rajesh Kumar. 2019. Design and exploration of load balancers for fog computing using fuzzy logic. *Simulation Modelling Practice and Theory* 101 (2019), 102017.

Olena Skarlat, Matteo Nardelli, Stefan Schulte, Michael Borkowski, and Philipp Leitner. 2017. Optimized IoT service placement in the fog. *Service Oriented Computing and Applications* 11, 4 (Dec. 2017), 427–443.

Z. Su, Q. Xu, J. Luo, H. Pu, Y. Peng, and R. Lu. 2018. A secure content caching scheme for disaster backup in fog computing enabled mobile social networks. *IEEE Transactions on Industrial Informatics* 14, 10 (Oct. 2018), 4579–4589.

Huaiying Sun, Huiqun Yu, Guisheng Fan, and Liqiong Chen. 2019. Energy and time efficient task offloading and resource allocation on the generic IoT-fog-cloud architecture. *Peer-to-Peer Networking and Applications* 13 (July 2019), 548–563.

M. Suter, R. Eidenbenz, Y. Pignolet, and A. Singla. 2019. Fog application allocation for automation systems. In *Proceedings of the 2019 IEEE International Conference on Fog Computing (ICFC'19)*. 97–106.

Madiha H. Syed, Eduardo B. Fernandez, and Mohammad Ilyas. 2016. A pattern for fog computing. In *Proceedings of the 10th Travelling Conference on Pattern Languages of Programs (VikingPLoP'16)*. ACM, New York, NY, Article 13, 10 pages.

Shreshth Tuli, Redowan Mahmud, Shikhar Tuli, and Rajkumar Buyya. 2019. FogBus: A blockchain-based lightweight framework for edge and fog computing. *Journal of Systems and Software* 154 (2019), 22–36.

Dimitrios Tychalas and Helen Karatza. 2020. A scheduling algorithm for a fog computing system with bag-of-tasks jobs: Simulation and performance evaluation. *Simulation Modelling Practice and Theory* 98 (2020), 101982.

Minoru Uehara. 2018. *Mist Computing: Linking Cloudlet to Fogs*. Springer International Publishing, Cham, Switzerland, 201–213.

P. Varshney and Y. Simmhan. 2017. Demystifying fog computing: Characterizing architectures, applications and abstractions. In *Proceedings of the 2017 IEEE 1st International Conference on Fog and Edge Computing (ICFEC'17)*. 115–124.

Salvatore Venticinque and Alba Amato. 2019. A methodology for deployment of IoT application in fog. *Journal of Ambient Intelligence and Humanized Computing* 10, 5 (May 2019), 1955–1976.

N. Verba, K. Chao, J. Lewandowski, N. Shah, A. James, and F. Tian. 2019. Modeling Industry 4.0 based fog computing environments for application analysis and deployment. *Future Generation Computer Systems* 91 (2019), 48–60.

Paola G. Vinueza Naranjo, Enzo Baccarelli, and Michele Scarpiniti. 2018. Design and energy-efficient resource management of virtualized networked fog architectures for the real-time support of IoT applications. *Journal of Supercomputing* 74, 6 (June 2018), 2470–2507.

Duc-Nghia Vu, Nhu-Ngoc Dao, Yongwoon Jang, Woongsoo Na, Young-Bin Kwon, Hyunchul Kang, Jason J. Jung, and Sungrae Cho. 2019. Joint energy and latency optimization for upstream IoT offloading services in fog radio access networks. *Transactions on Emerging Telecommunications Technologies* 30, 4 (2019), e3497.

D. Wang, Z. Liu, X. Wang, and Y. Lan. 2019a. Mobility-aware task offloading and migration schemes in fog computing networks. *IEEE Access* 7 (2019), 43356–43368.

T. Wang, J. Zhou, A. Liu, M. Z. A. Bhuiyan, G. Wang, and W. Jia. 2019. Fog-based computing and storage offloading for data synchronization in IoT. *IEEE Internet of Things Journal* 6, 3 (June 2019), 4272–4282.

Wei Wang, Guanyu Wu, Zhe Guo, Liang Qian, Lianghui Ding, and Feng Yang. 2019. Data scheduling and resource optimization for fog computing architecture in Industrial IoT. In *Distributed Computing and Internet Technology*, G. Fahrnberger, S. Gopinathan, and L. Parida (Eds.). Springer International Publishing, Cham, Switzerland, 141–149.

X. Wang, L. Wang, Y. Li, and K. Gai. 2018. Privacy-aware efficient fine-grained data access control in Internet of Medical Things based fog computing. *IEEE Access* 6 (2018), 47657–47665.

Y. Wang, K. Wang, H. Huang, T. Miyazaki, and S. Guo. 2019b. Traffic and computation co-offloading with reinforcement learning in fog computing for industrial applications. *IEEE Transactions on Industrial Informatics* 15, 2 (Feb. 2019), 976–986.

Mohammad Wazid, Ashok Kumar Das, Neeraj Kumar, and Athanasios V. Vasilakos. 2019. Design of secure key management and user authentication scheme for fog computing services. *Future Generation Computer Systems* 91 (2019), 475–492.

P. Wiener, P. Zehnder, and D. Riemer. 2019. Towards context-aware and dynamic management of stream processing pipelines for fog computing. In *Proceedings of the 2019 IEEE 3rd International Conference on Fog and Edge Computing (ICFEC'19)*. 1–6.

C. Wu and L. Wang. 2019. A deadline-aware estimation of distribution algorithm for resource scheduling in fog computing systems. In *Proceedings of the 2019 IEEE Congress on Evolutionary Computation (CEC'19)*. 660–666.

Y. Xiao and M. Krunz. 2018. Distributed optimization for energy-efficient fog computing in the Tactile Internet. *IEEE Journal on Selected Areas in Communications* 36, 11 (Nov. 2018), 2390–2400.

M. Yannuzzi, R. Milito, R. Serral-Gracia, D. Montero, and M. Nemirovsky. 2014. Key ingredients in an IoT recipe: Fog computing, cloud computing, and more fog computing. In *Proceedings of the 2014 IEEE 19th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD'14)*. 325–329.

J. Yao and N. Ansari. 2019. QoS-aware fog resource provisioning and mobile device power control in IoT networks. *IEEE Transactions on Network and Service Management* 16, 1 (March 2019), 167–175.

Y. Yao, X. Chang, J. Misic, and V. Misic. 2019. Reliable and secure vehicular fog service provision. *IEEE Internet of Things Journal* 6, 1 (Feb. 2019), 734–743.

L. Yin, J. Luo, and H. Luo. 2018. Tasks scheduling and resource allocation in fog computing based on containers for smart manufacturing. *IEEE Transactions on Industrial Informatics* 14, 10 (Oct. 2018), 4712–4721.

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* 98 (2019), 289–330.

A. Yousefpour, G. Ishigaki, R. Gour, and J. P. Jue. 2018. On reducing IoT service delay via fog offloading. *IEEE Internet of Things Journal* 5, 2 (April 2018), 998–1010.

L. Yu, T. Jiang, and Y. Zou. 2017. Fog-assisted operational cost reduction for cloud data centers. *IEEE Access* 5 (2017), 13578–13586.

J. Yue, M. Xiao, and Z. Pang. 2018. Distributed fog computing based on batched sparse codes for industrial control. *IEEE Transactions on Industrial Informatics* 14, 10 (Oct. 2018), 4683–4691.

W. Yanez, R. Mahmud, R. Bahsoon, Y. Zhang, and R. Buyya. 2020. Data allocation mechanism for Internet-of-Things systems with blockchain. *IEEE Internet of Things Journal* 7, 4 (2020), 3509–3522. DOI: https://doi.org/10.1109/JIOT.2020.2972776

Deze Zeng, Lin Gu, and Hong Yao. 2018. Towards energy efficient service composition in green energy powered cyber-physical fog systems. *Future Generation Computer Systems*.

M. Zeng, Y. Li, K. Zhang, M. Waqas, and D. Jin. 2018. Incentive mechanism design for computation offloading in heterogeneous fog computing: A contract-based approach. In *Proceedings of the IEEE International Conference on Communications (ICC'18)*. 1–6.

G. Zhang, F. Shen, N. Chen, P. Zhu, X. Dai, and Y. Yang. 2019a. DOTS: Delay-optimal task scheduling among voluntary nodes in fog networks. *IEEE Internet of Things Journal* 6, 2 (April 2019), 3533–3544.

G. Zhang, F. Shen, Z. Liu, Y. Yang, K. Wang, and M. Zhou. 2019b. FEMTO: Fair and energy-minimized task offloading for fog-enabled IoT networks. *IEEE Internet of Things Journal* 6, 3 (June 2019), 4388–4400.

PeiYun Zhang, MengChu Zhou, and Giancarlo Fortino. 2018. Security and trust issues in fog computing: A survey. *Future Generation Computer Systems* 88 (2018), 16–27.

Dongcheng Zhao, Gang Sun, Dan Liao, Shizhong Xu, and Victor Chang. 2019. Mobile-aware service function chain migration in cloud–fog computing. *Future Generation Computer Systems* 96 (2019), 591–604.

S. Zhao, Y. Yang, Z. Shao, X. Yang, H. Qian, and C. Wang. 2018. FEMOS: Fog-enabled multitier operations scheduling in dynamic wireless networks. *IEEE Internet of Things Journal* 5, 2 (April 2018), 1169–1183.

H. Zheng, K. Xiong, P. Fan, Z. Zhong, and K. B. Letaief. 2019b. Fog-assisted multiuser SWIPT networks: Local computing or offloading. *IEEE Internet of Things Journal* 6, 3 (June 2019), 5246–5264.

Q. Zheng, J. Jin, T. Zhang, J. Li, L. Gao, and Y. Xiang. 2019a. Energy-sustainable fog system for mobile web services in infrastructure-less environments. *IEEE Access* 7 (2019), 161318–161328.

Jingya Zhou, Jianxi Fan, Jin Wang, and Juncheng Jia. 2019. Dynamic service deployment for budget-constrained mobile edge computing. *Concurrency and Computation: Practice and Experience* (2019), e5436. DOI: https://doi.org/10.1002/cpe.5436

Z. Zhou, P. Liu, J. Feng, Y. Zhang, S. Mumtaz, and J. Rodriguez. 2019. Computation resource allocation and task assignment optimization in vehicular fog computing: A contract-matching approach. *IEEE Transactions on Vehicular Technology* 68, 4 (April 2019), 3113–3125.

C. Zhu, J. Tao, G. Pastor, Y. Xiao, Y. Ji, Q. Zhou, Y. Li, and A. Yia-Jaaski. 2019. Folo: Latency and quality optimized task allocation in vehicular fog computing. *IEEE Internet of Things Journal* 6, 3 (June 2019), 4150–4161.

Received January 2020; revised April 2020; accepted May 2020