Live Virtual Machine Migration in Fog Computing: State of the Art

Shahd Alqam
Computer Science Department,
Sultan Qaboos University Muscat, Sultanate of Oman
S125248@student.squ.edu.om

Nasser Alzeidi
Computer Science Department,
Sultan Qaboos University Muscat, Sultanate of Oman
alzidi@squ.edu.om

Abderrezak Touzene
Computer Science Department,
Sultan Qaboos University Muscat, Sultanate of Oman
touzene@squ.edu.om

Khaled Day
Computer Science Department,
Sultan Qaboos University Muscat, Sultanate of Oman
kday@squ.edu.om

Abstract: *Fog computing is an emerging paradigm which extends the functionality of the cloud near to the end users. Its introduction helped in running different real-time applications where latency is a critical factor. This paradigm is motivated by the fast growth of Internet of Things (IoT) applications in different fields. By running Virtual Machines (VMs) on fog devices, different users will be able to offload their computational tasks to fog devices to get them done in a smooth, transparent, and faster manner. Nevertheless, the performance of real-time applications might suffer if no proper live virtual machine migration mechanism is adopted. Live VM migration aims to move the running VM from one physical fog node to another with minimal or zero downtime due to mobility issues. Many efforts have been made in this field to solve the challenges facing live VM migration in fog computing. However, there are remaining issues that require solutions and improvements. In this paper, the following presents the research outcomes: An extensive survey of existing literature on live VM migration mechanisms in fog computing. Also, a new novel classification approach for categorizing live VM migration mechanisms based on conventional and Artificial Intelligence (AI) approaches to address live VM migration challenges is presented. Moreover, an identification of research gaps and in the existing literature and highlighting the areas where further investigation is required is done and finally a conclusion with a discussion of potential future research directions is drawn.*

Keywords: *Live virtual machine migration, conventional algorithms, fog computing architecture, live migration algorithms classification, artificial intelligence, framework modeling.*

Received June 16, 2023; accepted October 5, 2023
https://doi.org/10.34028/iajit/20/6/14

1. Introduction

Although there are many definitions in the literature for Internet of Things (IoT); it can be defined as an emerging technology to connect different physical things "objects" over the Internet [1]. It was first introduced when Kevin Ashton used the term for the first time in 1999 to talk about internet-based information service architecture [2]. According to Ahmed *et al*. [3], IoT is the integration of mobile networks, Internet, social networks, and things which are connected to provide different services and applications to users. IoT has given birth to many services and applications which have a huge impact on humans' daily life. This includes smart cities, smart environment, smart farming, smart healthcare, smart grid, Augmented Reality (AR), etc. IoT has gained importance being successfully implemented in different fields and areas [4]. Some of these applications are time-critical where the latency is required to be minimal as most of the interactions are expected to be done in real time [5].

Cloud computing supplied IoT applications with unlimited storage and processing capabilities. The cloud has also facilitated the interactions between the "things," which made IoT and its applications a reality [6]. However, although cloud computing has been an effective way to fulfill the computational and storage requirements for IoT applications, it might not be suitable option for certain types of real-time applications to fulfill their latency requirements [7]. Latency-critical applications such as AR, VR, video surveillance, to name a few, cannot wait for the response to be sent back from faraway cloud data centers. These latency requirements are hard to achieve with existing cloud computing architectures. Moreover, the applications running on IoT produce a massive volume of data that needs to be transferred through the network and then processed. The amount of data generated from the 'things' is continuously growing [6]. Therefore, it would not be possible and not feasible to send all the data collected from IoT devices (cameras, sensors, smartphones, etc.,) to the cloud all the time. That would incur high costs in terms of bandwidth, storage, latency, and energy.

There was a need to resolve these weaknesses of the

cloud computing and introduce a middle layer between the cloud and the "things" to meet the extreme latency requirements of the real-time applications [8] and hence, fog computing as a new paradigm has evolved [9]. Fog computing is an architecture that uses one or more near-edge devices to carry out some amount of storage, communication, control, configuration, and management of the cloud [9]. It does not substitute the cloud but extends its functionality near to the edge users. Fog computing has been introduced to be an intermediate layer between the IoT devices and the cloud layer [10].

Considering different factors like the mobility of the users, QoS, service cost, and the capabilities of fog nodes, there is a need to be able to offload some tasks from one fog node to another to achieve higher flexibility and adaptability [11]. Researchers Bi *et al*. [12] have proposed to migrate fog-based Virtual Machines (VMs) for a variety of reasons such as load balancing, users' mobility, priorities, etc.

VM migration normally refers to the hardware-level migration where the VM share the hardware resources and can run different Operating Systems (OSs) platforms. It applies two techniques: Cold migration and live migration [13]. In cold migration the source VM is frozen and suspended for some time, then it is transferred to the destination and finally resumed on the destination and removed from the source node [13]. Cold migration might result in a very long downtime, which goes against the main objective of migrating the application/service to support real-time application with minimal latency. On the other hand, live VM migration transfers the VM from one node to another with minimal disruption time [14]. It strives to provide the end user with the service without interruption during the migration process [15].

In this paper, the state of the art of live VM migration mechanisms in fog computing is critically evaluated and analyzed. The contributions of this paper are three-fold:

- First, a comprehensive review and analysis of the existing live VM migration mechanisms is presented.
- Second, light is shed on existing classifications of these mechanisms and propose a new one.
- Finally, this paper presents the research gaps and highlights some directions for further research studies.

The rest of this paper is organized as follows: Section 2 provides some background information and preliminaries about fog computing and virtualization. Section 3, presenting a detailed literature review and analysis of the existing live VM migration proposals. In section 4, a classification of these tools and propose a new classification is discussed. Research challenges and directions are then outlined in section 5. The paper concludes with final remarks about migration in container-based systems in section 6.

2. Preliminaries and Background

2.1. Characteristics of Fog Computing

- Multi-Protocol Support: Fog computing is designed in such a way to support many communication protocols. These include: Wi-Fi, ZigBee, Bluetooth, Advanced and Adaptive Network Technology (ANT), Z-wave, 6 Low-power Wireless Personal Area Network (6 LowPAN), Thread, Worldwide Interoperability for Microwave Access (WiMAX), Low-Power Wide Area Network (LPWAN), Cellular, Radio Frequency Identifier (RFID), Near Field Communication (NFC), HTTP, Extensible Messaging Presence Protocol (XMPP), Message Queuing Telemetry Transport (MQTT), Constrained Application Protocol (CoAP), Advanced Message Queuing Protocol (AMQP), IoTivity, Hyper-Cat, AllJoyn, etc., [9].
- Mobility and Geo-Graphical Distribution: Mobility is the main feature of fog computing [9]. IoT devices and fog devices can be mobile and therefore, fog systems should be able to handle and support this mobility [16]. Mobility is considered as the main characteristic of many smart city applications. Mobility in fog computing allows the fog devices to manage a large number of IoT devices across different geographical areas [9]. Further, the services and applications provided are distributed and can be deployed anywhere in the fog environment [17].
- Low-latency: One of the characteristics of fog computing which is considered the main driver behind introducing fog computing is latency-awareness. It aims at offering the end users with guaranteed low-latency applications [18]. Factors including latency in fog computing include execution time, task offloading time, and transmission time.
- Heterogeneity: Fog computing is designed in such a way to support a wide range of heterogonous devices [18]. In the IoT architecture, the bottom layer consists of various types of edge devices with different software and hardware architectures and specifications. It is heterogeneous even in terms of the fog nodes and the network infrastructure. This is because the fog layer might include high-end servers, routers, wireless access points, proxy servers, end-devices like vehicles, mobile phones, base stations, etc. These devices are heterogeneous in terms of hardware and software architecture and capabilities.

2.2. Applications and Use Cases of Fog Computing

Fog computing was initially designed and introduced to support latency-sensitive applications. This includes smart cities applications, ITS, AR, healthcare, tele-surveillance, smart grid, smart agriculture, smart waste management, smart water management, smart retail store, etc., [18, 19, 20, 21]. The following illustrates

some of these applications:

- Smart City: the city in this context refers to the urban areas which require all things to be smart. To build and construct a smart city, it is needed to ensure real-time responses, latency-aware applications and location-aware services [19]. Fog plays a major role in supporting these various types of applications of smart cities including noise pollution reduction, urban drainage networks, smart buildings, etc., [20].
- Intelligent Transportation System (ITS): ITS is one of the main components of smart cities. ITS systems generate a huge amount of data which make it difficult to be transferred to the cloud [19]. Further, ITS applications such as smart vehicles are time-critical and require fast response to ensure safety. This could be done by integrating the ITS with Fog computing [19]. As known, driving vehicles in cities require immediate decision and response to different cases and events. For example: an immediate decision is required in case of slowing speed down, sudden traffic, lane change, route change, blocked routes, etc., [20].
- Healthcare System: fog Computing is applied in healthcare for monitoring, detection, diagnosis and visualization of the health status of people [21]. It plays a vital role as many healthcare related issues need to be monitored in real-time. Applications in healthcare are latency sensitive. They need to send real-time data to monitor the status of the patients. [16].
- Augmented Reality: The applications under AR are extra time-sensitive where a microsecond delay makes a difference. A small delay might cause a fatal error and hence, fog computing is adopted in this field [20].

2.3. Fog Computing Architecture

The term architecture has different meanings across the literature. This section presents some of the most common architectures of fog computing:

1. Building architecture: this is concerned with engineering principles of the fog layer and the properties of the materials [22].
2. Hardware architecture: this describes the configurations of the components making up the fog node hardware and the working principles including the operations of the processors, etc. It is concerned also with the laws of physics.
3. Software architecture: this defines the OSs and related software used in the fog nodes [22].

However, the most famous and common way to get a system architecture of fog computing is to have a layered architecture. In fog computing, the layered architecture is concerned with the components which are organized into a set of layers on top of one another. The idea is that the components in each layer will provide services and details to the upper layers, and it may abstract some details to the higher layers.

The following are some existing layered architectures of fog computing. Yet, there is no standard architecture followed in the fog computing field.

- OpenFog Consortium: in this architecture, the fog layer covers all the computing facilities between the end devices and the cloud servers. The fog nodes may connect using wireless or wired communication. Each layer differs in terms of processing capabilities, network, etc., [23]. Figure 1 represents the OpenFog layered architecture.

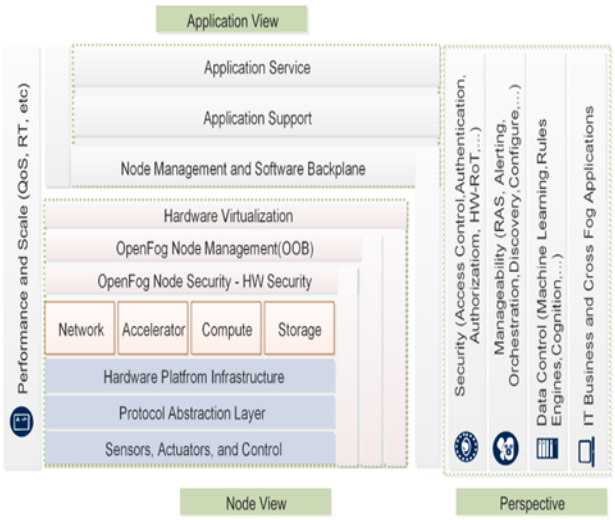

Figure 1. OpenFog layered architecture [23].

- Cisco Bonomi Architecture**:** This architecture was created and developed by Flavio Bonomi. He designed this architecture with consideration of the heterogeneity of fog nodes. The architecture consists of five main components as shown in Figure 2: The heterogeneous physical resources layer, the fog abstraction layer, the fog services orchestration layer, the fog abstraction layer, IoT services, and distributed message bus [23]. As per this architecture, the heterogeneous physical resource layer consists of servers, edge routers, access points, vehicles, mobile phones, etc., [23].

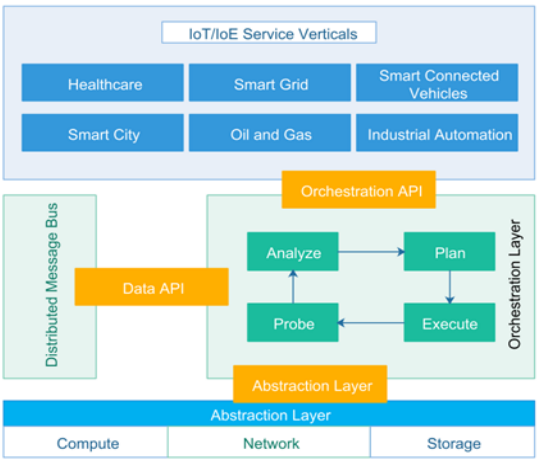

Figure 2. Cisco bonomi architecture [23].

- Three-layer Architecture: One of the fog computing architectures is the 3-layers architecture as illustrated in Figure 3. It consists of terminal, fog, and cloud layers.
  - Terminal layer: this layer is the closest layer to the end users and physical environment. It consists of sensors, mobile phones, smart vehicles, readers, etc. They sense and collect the data and send it to the upper layer which is the fog layer.
  - Fog layer: this layer is composed of a large number of heterogeneous fog nodes including routers, switches, proxy servers, bus stations, gateways, dedicated fog servers, etc. These devices can be found in a distribution fashion. They can be found in cafes, shopping centers, streets, parks, etc. They can be static or mobile.
  - Cloud layer: it consists of multiple high-performance servers, [23].

Table 1 summarizes the key concepts of fog computing.

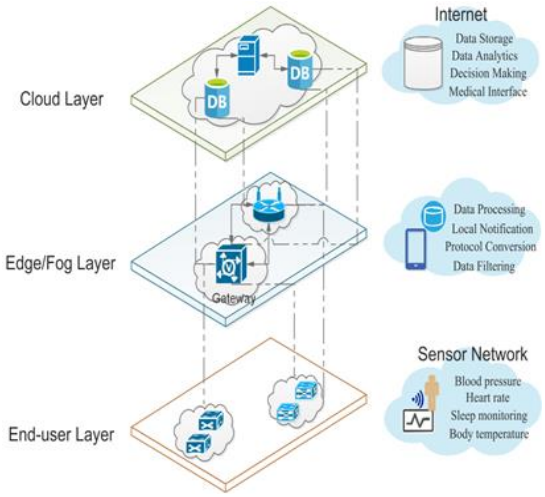

Figure 3. Three-layers architecture [23].

Table 1. Summary of the key concepts of fog computing.

Sub-sections	Key concepts	Description	Examples
Characteristics of fog computing.	Multi-Protocol Support	Fog computing supports wide range of dissimilar protocols	Wi-Fi, ZigBee, Bluetooth, ANT, Z-wave, 6 LowPAN, etc.
	Mobility and Geo-Graphical Distribution	It is supported and considered as the main feature of fog computing	Manage a large number of IoT devices across different geographical areas.
	Low-Latency	Considered as the main trigger to create the fog computing	Required in time-critical applications like VR, AR etc.
	Heterogeneity	Fog node function could be built on any type of machines.	Include high-end servers, routers, wireless access points, proxy servers, etc.
Applications and use cases of fog computing.	Smart City	Fog is needed to ensure real-time responses, latency-awareness.	Noise pollution reduction, urban drainage networks, smart buildings, etc.
	ITS	Generate a huge amount of data of time-critical application which require fast and prompt response	An immediate decision is required in case of slowing speed down, sudden traffic, etc.
	AR	An extra time-sensitive. Hence, fog computing is required	AR in medical sector, AR in education, etc.
	Healthcare system	Applications are latency sensitive.	Monitoring, detection, diagnostics.
Fog computing architecture.	Building architecture	Concerned with engineering principles of the fog layer and the properties of the materials	-
	Hardware architecture	Describes the configurations of the components making up the fog node hardware	Processors
	Software architecture	Defines the OSs and related software	OS, applications
	OpenFog consortium	The fog layer covers all the computing facilities between the end devices and the cloud servers.	
	Cisco bonomi architecture	Five main components: The heterogeneous resources layer, the fog layer, the fog services layer, the fog abstraction layer, IoT services	
	Three-layers architecture	Terminal layer, fog layer, cloud layer.	

3. Fog Computing and Resource Management

3.1. Virtualization and Resource Management in Fog Computing

The real-time applications running in IoT generate many tasks with stringent delay requirements. However, considering few factors related to the fog nodes, such as the mobility of the users connected to fog nodes, the low-resources available in fog nodes, heterogeneity of fog nodes, etc., the tasks may have to be migrated from one fog node to other fog nodes in order to meet the time requirements [11].

One of the significant issues in resource allocation in fog computing is offloading. Offloading in fog means that, if a user is running a task and the resources are not enough in the assigned fog node, it will offload the task to the cloud or to a nearby fog node with higher computational capability [24].

In cloud/fog computing, resource allocation is a concept of allocating and assigning the free and available resources to the IoT users over the Internet. The resources could be either physical resources or virtual resources [25]. Different parameters need to be considered while allocating the resources such as latency requirements, throughput, energy consumption, etc. However, by applying and using virtualization, the resources could be better utilized in the fog layer.

Virtualization can be defined as a technique for abstracting the computational resources such as processors, memory, storage, I/O devices, etc. to provide virtualized resources to the IoT users through

VM [25]. Virtualization creates multiple virtual environments on a single physical device which allows to serve many users and process many tasks simultaneously [26]. Virtualization is considered as the engine that drives cloud computing and enables it to exist. It turned the data centers into a self-managed, highly scalable, and highly available pool of resources. So, by dividing and abstracting the physical hardware into logical divisions VMs, resource management became more efficient in cloud computing and hence the same applies to fog computing.

3.2. How Virtualization Works

The virtual environment is controlled and managed by software called Hypervisor. The hypervisor is a layer of software that resides between the hardware of the device and the VM of the node.

The hypervisor is important as without having it, the OS will communicate directly with the hardware and hence, more than one OS from different VMs would simultaneously try to control the hardware which results in chaos and conflict. Hypervisor manages all these interactions between individual VMs and the hardware resources of the node. As in our research, VMs will be discussed, it is worth discussing the two types of hypervisors.

- Type 1 (bare metal) hypervisor: In this type, the hypervisor is located directly on top of the physical hardware. Because there is no extra layer between the hypervisor and the hardware, it is called a bare-metal type. It is directly communicating with the hardware resources [14]. Figure 4 illustrates type 1.

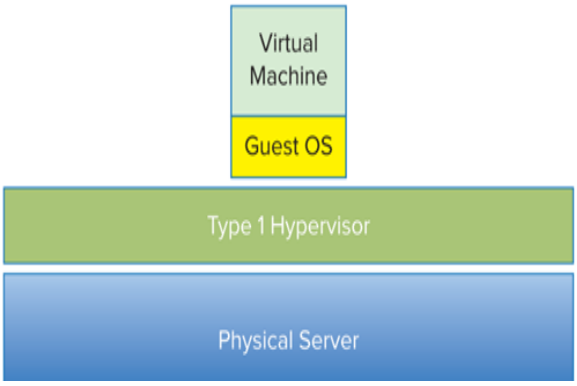

Figure 4. Bare-metal hypervisor (virtualization essentials, n.d.).

- However type 2 hypervisor is different in that the hypervisor software is not installed directly on the physical hardware but instead runs on the top of the device's OS as illustrated in Figure 5. It shares the OS libraries and creates guest OSs instead of guest VMs [14].

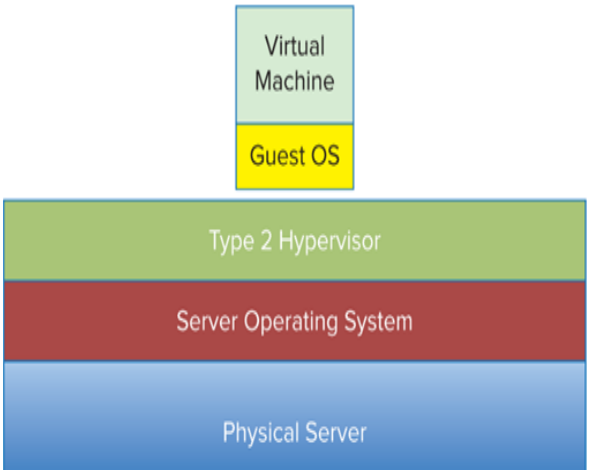

Figure 5. Type 2-hosted hypervisor [27].

3.3. Virtual Machine Migration

VM Migration in fog computing is defined as transferring the VM from source a fog node to another more appropriate fog node. This could be done because of several reasons. First, users in IoT scenarios are moving very frequently which requires migration of VM. Secondly, when a fog node cannot handle more requests and it is overloaded; it is better to offload to another fog node [12].

VM migration in fog computing enables resource allocation efficiently. However, the migration of VM in the fog layer is influenced by different aspects that are not present in cloud computing [13]. The main challenges when discussing VM migration in fog computing are:

1) High heterogeneity of fog nodes. This means that the fog layer consists of different nodes with different hardware architecture, capabilities, and types. The virtualization process should consider heterogeneity while adopting a VM migration solution.
2) Fog nodes are connected using Wide Area Network connections with different communication systems; hence, huge latency and delay may result.
3) In cloud computing, the TMT is a secondary concern while in fog computing it is a major concern.
4) The fog nodes are not dedicated for fog computing purposes only. They are busy with their main functions. For example, the fog can be built on the router where the router's main priority is to find the best route in the network rather than fog-related tasks [20].

There are two types of VM Migration: Cold VM migration and live VM migration as presented in Figure 6. The following section provides more details on both types of VM migration.

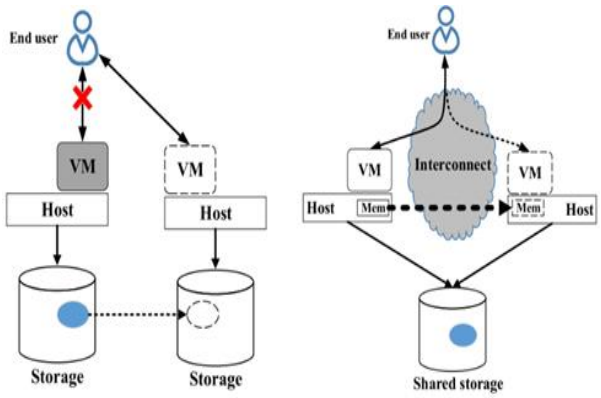

Figure 6. Live vs. non-live migration [28].

3.3.1. Cold (Non-Live) VM Migration

In this type of migration, the source VM is frozen and suspended for some time, then it is transferred to the destination and finally resumed on the destination and removed from the source node [13].

According to Jiang *et al.* [28], no open network connection is kept during the non-live (cold) migration and all connections are rebuilt only after VM resumption. The main issue in this type is the downtime. The downtime is the duration in which the migrated VM is out of service. However, non-live migration cannot be adopted in fog computing scenarios due to its long downtime which is detrimental to the requirements of real-time applications. Live migration was proposed initially to move the VMs between different physical servers without any dis-connectivity or downtime. Figure 7 represents this type of migration.

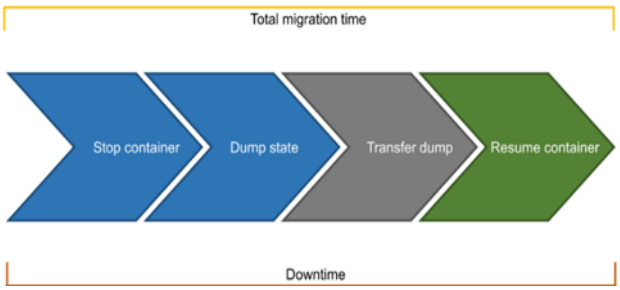

Figure 7. Cold migration [13].

3.3.2. Live VM Migration in Fog Computing

Live VM migration is a technique of transferring the VM from one node to another with minimal downtime [14]. It ensures service availability during the migration process [15]. It is illustrated in Figure 8. Live VM migration is used for many reasons as follows:

- Load balancing: When a machine is overloaded with user requests and many VMs are running on it, its performance is degraded and may become unable to process its tasks. Another machine may be underutilized at that same time, which will be wasteful for its capabilities. Live VM migration is an appropriate solution to transparently balance the load without affecting the running machines and without the user noticing the distribution of the service [15, 28].
- Resource sharing: Migration can be used when multiple physical nodes have limited resources and need to work together as a single machine to share their resources. Relocating the VMs would make it possible to share the resources between multiple machines. This is referred to as a consolidation of resources [15].
- Energy saving: When different machines work below the normal load and are idle for most of the time, some technique is required to switch them off and save energy. This would be possible with VM migration [15].
- Preserving service availability: When running real-time applications, it is needed to ensure continuous service and availability for the end users. Live VM migration can serve this purpose.

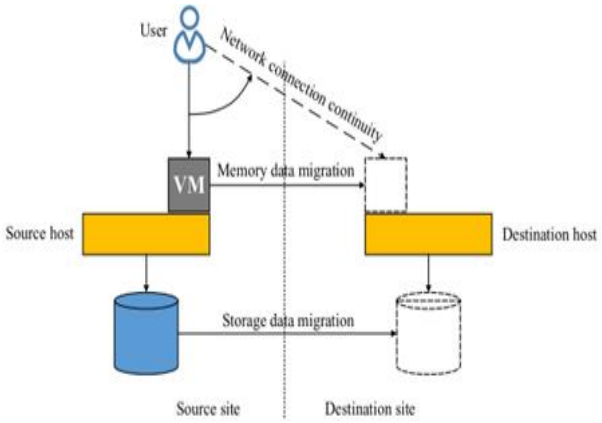

Figure 8. Live migration [12].

During live VM migration, the source machine keeps running while its files, states, disk storage and connections are being transferred to the destination machine [13]. Live VM migration can be performed across geographically distributed fog nodes. There are three components which need to be considered in live VM migration: memory state migration, storage migration and network connection migration [29].

According to Kaur *et al.* [30], live VM migration consists of three main phases: the push phase, stop-and-copy phase and pull phase. In the push phase, the virtual machine to be migrated continues running on the source physical machine while its memory pages are being transferred to the destination machine. In the stop and copy phase, the source VM is suspended, and the remaining dirty and un-transferred memory pages are copied to the destination. Finally, in the pull phase the virtual machine is totally resumed on the destination physical machine and removed from the source machine. The live VM migration involves a dynamic transfer from the source physical (hosting) machine to the destination machine. There are some key performance metrics to consider while designing a live migration solution. The following lists some of those performance metrics:

- Total Migration Time (TMT): It is the total duration from the start of the migration process until the virtual machine is completely resumed on the destination machine and removed from the source machine. This duration is impacted by multiple factors including:
  1) The bandwidth and speed of the network links.
  2) The size of the VM to be migrated.
  3) The speed of the I/O Operations.
  4) The size of the memory pages, etc., [31]. It is estimated as follows:

$$TMT = \frac{total\ memory\ used\ by\ VM}{bandwidth} \quad (1)$$

- Downtime: It is the total time during which the VM is unavailable to the user during the second (stop-and-copy) phase of the migration until it resumes on the destination machine [31]. It is calculated as follows:

$$DT = \frac{fd * p * tn}{s} \quad (2)$$

where *fd* is the number of dirty pages of the memory, *p* is the page size, *tn* is the duration of the copy function and *s* is the link speed [31].

- Link Speed: It is the capacity of a link, or the bandwidth allocated to the link. It is inversely proportional to the downtime and to the TMT. This is clearly captured in the above equations [31].
- Total network traffic: This metric is also important. It is the amount of data transferred across the network connection from the source to the destination [31, 32].

3.4. Issues in Live Migration

Migration of VMs in the fog layer is influenced by different aspects that are not present in cloud computing [13]. The main challenges when discussing VM migration in fog computing are:

- High heterogeneity of fog nodes: The fog layer consists of different nodes with different hardware architectures, capabilities, and types. The fog environment is characterized by a high level of node heterogeneity. The VM migration process should take into consideration this heterogeneity. It should function on different types of fog nodes with different specifications and architectures [13].
- Connection of fog nodes: Fog nodes are usually connected via wireless links with different communication systems that are subject to long latency and throughput. Heterogeneity is also observed in terms of network topologies and connections (e.g., Wi-Fi, Long Term Evaluation (LTE), 4G, 5G, etc.,) [26].
- Functionality of the fog nodes: The fog nodes are not dedicated for fog computing purposes. They have their own main functions. For example, the fog can be built on the router where the router's main priority is to find the best route in the network and the fog-related tasks will be the second priority [20].

Table 2 summarizes the key findings of VM migration.

Table 2. Summary of the key findings of VM migration.

Sub-sections	Key Findings
Virtual machine migration	The aspects which influence the migration of VMs in the fog layer: • High heterogeneity of fog nodes. • Wide Area Network connections with different communication systems. • The TMT is a major concern. • The fog nodes are not dedicated for fog computing purposes only.
	Live VM migration is used for many reasons: • Load balancing. • Resource sharing. • Energy saving. • Preserving service availability.
	The key performance metrics to consider while designing a live migration solution. • TMT. • Downtime. • Link speed. • Total network traffic.
Issues in live migration	• High heterogeneity of fog nodes. • Connection of fog nodes. • Functionality of the fog nodes.

4. A Classification of Live VM Migration Methods

There are various types of live virtual machine migration methods in the fog-computing environment. The following provides a classification of the existing methods. It is worth noting that this is the first classification in literature that classifies the existing methods. The proposed classification classifies the existing solutions based on algorithm and framework modeling. It further broken into subcategories, where algorithm-based classification is divided into conventional based and Artificial Intelligence (AI) based. The Figure 15 details the new classification of the existing solutions.

There are some differences between algorithm-based and framework modelling-based solutions. A framework modelling-based solution provides only guidelines to follow. It is a scheme for applying ready software or components. On the other hand, an algorithm-based solution provides a systematic approach for performing live VM migration in a fog-computing environment.

There are two categories of algorithm-based solutions. Traditional/conventional algorithms and AI-based algorithms. AI-based solutions are more suitable for solving complex and specialized problems where the dynamic and large environment is a challenge. On the other hand, a traditional algorithm solution is more suitable for broad problems. Further, a traditional algorithm results in static output, which makes it inappropriate to cope with dynamically changing

environments like fog computing and mobile computing. In dynamic environments, the learning will be more efficient when the system learns from its own experience.

5. Evaluation and Analysis of Existing Live VM Migration Solutions

In this section, an evaluation and analysis of existing live VM migration techniques out of both types of algorithm-based and framework modeling-based is presented.

5.1. Algorithm Based Solutions

5.1.1. Conventional Algorithms Based

5.1.1.1. Bin Packing Algorithm

Author Mann in [33] addressed load balancing in fog/cloud environment. They discussed where to place the VMs according to a Heuristic VM Scheduling Strategy. The problem of minimizing the load balance variance for the fog nodes was formulated as:

$$\sum_{n=1}^{N} I_n^m r_n \leq \theta_m \quad (3)$$

where $\sum_{n=1}^{N} I_n^m r_n$ represents the resources allocated to type m of computing nodes and θm is the amount of VM instances of that type. The applied strategy allocates the VMs to the computing nodes which have the same node type as the required VM instance type, and which have enough spare space to host the VM. This proposal applied Bin packing Best Fit Decreasing (BFD) which is applied in scenarios where there is a need to realize the process of computing nodes which assign the VM into appropriate node. An experimental evaluation and comparison analysis were conducted to validate the efficiency of the proposed algorithm. A comparison was made between First Fit Decreasing (FFD), BFD and the suggested Heuristic Scheduling Method (HSM). The results showed better performance and better resource utilization when applying the proposed HSM in comparison to FFD and BFD.

Although the study shows better performance compared to other studies, it did not consider the heterogeneity between the nodes of specific type (Fog/Cloud). It assumes that all nodes of specific type are homogenous and have the same architecture and resources. Further, the study considers the users are in a stationary mode.

5.1.1.2. Integer Linear Programming

A different approach is used in the study [34] where the proposal used Integer Linear Programming (ILP) to find out the best fog device to migrate the VM to it. It considered mobility prediction and starts the process before 5 minutes of user movement. Mouradian *et al.* [34] used ILP to optimize the selection of physical machines. The objective functions are:

1) Maximizing the accepted requests.
2) Minimizing the latency.

$$max \sum_{a \in A} \sum_{n \in N} P_{a,n} \quad (4)$$

where A is a set of applications to be executed, N is a set of nodes and $P_{a,n}$ is the placement matrix element with value 1 if the application a is placed in the node n.

$$min \sum_{a \in A} \sum_{n \in N} P_{a,n} C_{a,n} \quad (5)$$

where $C_{a,n}$ is the cost matrix which calculates the latency between a node $n \in N$ and the user owner of the application $a \in A$.

The study considered the requirements of applications at the end user (edge) and how to choose the best fog node to process the application based on the application needs and computational resources of the nodes. It ensured that the application is executed on one server among all fog servers available. The server is selected based on the available resources. However, the limitations are: It assumes that all servers are of the same type but may have different computational resources. Further, in the simulation, they concentrate on the mobility aspect and assume all fog nodes have the same computational resources. The proposed model did not include any parameters about the mobility in the equations. It only considers mobility in the simulation by using ready model. However, ILP is better used in less complex scenarios where there is a set of predefined physical fog devices, and it is only requiring deciding where to migrate them.

5.1.1.3. Generic Fat Tree Algorithm

Authors Mukherjee *et al.* in [35] proposed an algorithmic model by selecting a generic fat tree architecture as an underlying topology. The main aim is to reduce the latency and provide redundant paths at any given time. The idea behind the fat tree is that it provides a full redundancy of the network wherein there are always available paths between any two switches at any given time as illustrated in Figure 9. It considered redundancy as a main requirement, which is a critical factor in live migration. The proposal consists of three layers: Edge layer is at the bottom of the system, aggregation layer is in middle of the system, and core layer is the upper one. Host H_h is the server layer that performs only two actions for sending and receiving the VMs. Mukherjee *et al.* [35] there are enough resources to allocate the VMs within the system. The lowest layer is the edge layer E_i, which is directly connected to the servers. The aggregation layer A_j is the middle layer where all switches monitor their ports all the time waiting for incoming VMs. The core layer C_L interconnects different pods so that the ports of all switches of this layer are looking downward to provide a full mesh topology among all existing pods. Although

this mechanism ensured availability and redundancy, it is not suitable for fog computing. It assumed that the fog network consists of switches connected with wires only, which does not match the fog characteristics. Further, it is a costlier solution and difficult to implement in the fog layer. Further, this study did not discuss how to decide migration considering computational and communication resources. It only considered the available paths. No evaluation was carried out to compare this mechanism with other solutions.

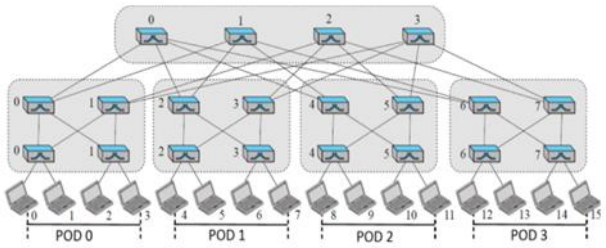

Figure 9. Fat tree topology [35].

5.1.1.4. Auction Algorithm

The authors Naha *et al*. in [36] implemented a strategy of computation offloading under a scenario of multi-users which considered the performance of the intelligent devices and servers as illustrated in Figure 10. In this proposal, Naha *et al*. [36] adopted an auction algorithm which takes into consideration the time requirements of the applications and the resources of the servers. Their mode is divided into three stages. The first stage decides whether there is a need to offload the tasks or not based on the energy and time consumption on the local mobile/edge devices. If it is decided to offload or migrate tasks, the second stage applies an Analytic Hierarchy Process (AHP) method to select the suitable server to offload based on the time and energy consumption and the CPU resources. In the third stage, the tasks are scheduled in the appropriate virtual machine by applying an improved auction algorithm. The time and energy consumptions are formulated as follows:

$$T_L = \frac{C_n}{V_L} \tag{6}$$

where C_n represents the computational resources required by the task n and V_L is the execution rate of the local CPU.

$$E_L = T_L \, x \, P_L \tag{7}$$

where P_L is the computing power of mobile devices. The condition for offloading is formulated as follows:

$$W > \frac{V_L P_{up} D_n}{P_L C_n \log_2(1 + \frac{P_{up} \, x \, Los}{N})} \tag{8}$$

Where W is the channel bandwidth, P_{up} is the upload power of the mobile devices, Los is the channel gain, N is the Gauss noise power in the channel and D_n is the amount of data that needs to be uploaded for the computing tasks.

The following represents the tasks' scheduling auction model: This proposal did not address when to migrate but only concentrates on where to migrate. It did not consider downtime or latency reduction. It only ensured the task was accomplished.

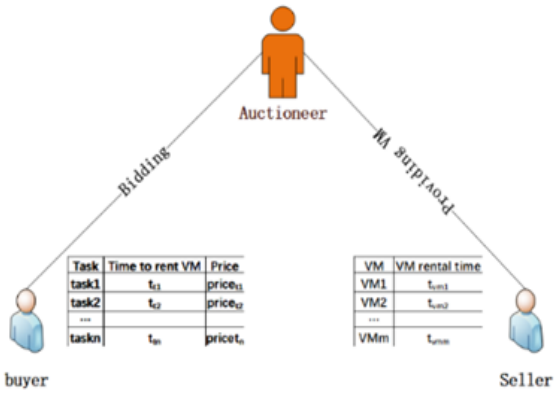

Figure 10. Auction model for tasks' scheduling [38].

5.1.1.5. Smart Elastic Scheduling Algorithm (SESA)

This study [37] aims to analyze and study existing techniques and focusing on SESA and Modified Best-Fit Decreasing as illustrated in Figure 11. First, the researchers classified the work into main sections. The first section deals with how clusters are organized within the nodes. Then, it focuses on placement of VMs considering the VMs that need to be migrated as per Modified Best-Fit Decreasing Algorithm. The model particularly emphasis is on addressing the power efficiency and resource utilization of VM placement.

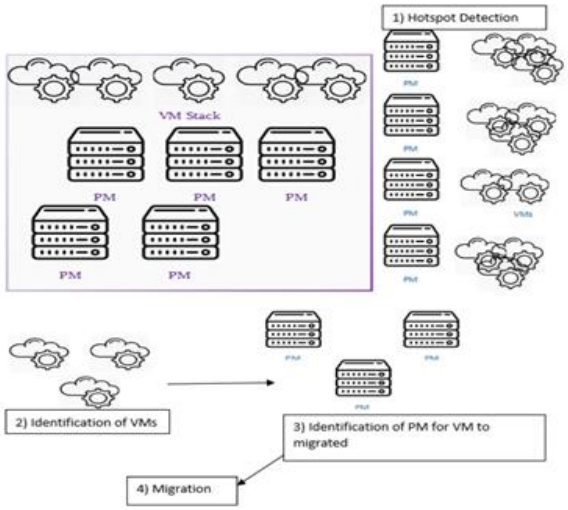

Figure 11. SESA algorithm [37].

5.1.1.6. A Service Migration Method for Resource Competition (SMRC)

This study [38] depends on the request/response aspect as shown in Figure 12. The users are considered mobile but with fixed route. The decision of migration depends on the user's movement speed and data transmission

rate. This study considers the migration from one base-station to another only. The task can send a pre-migration request and then the user can migrate the task.

A	When the base station to which the U_i is connected has changed, U_i reports it's information to E_{source}.
B	E_{source} consolidates all tasks that need to be migrated to the same $E_{candidate}$. The consolidated resource pre-request information is then sent to each $E_{candidate}$.
C	$E_{candidate}$ decides the amount of resources to be allocated to each E_{source} and sends the pre-allocated information to E_{source}. For pre-allocated resources, $E_{candidate}$ temporarily locks them.
D	E_{source} determines which tasks are actually migrated according to the priority of tasks. For user tasks that failed to migrate, change the best candidate server and try the migration again. Finally, E_{source} returns the number of unallocated resources to $E_{candidate}$.

Figure 12. A summary of the steps of SMRC [38].

5.1.2. AI- based Solutions

5.1.2.1. Deep Reinforcement Learning

Authors Perera *et al*. in [39] proposed a container migration architecture considering the communication delay and computational power. The proposal is implemented using reinforcement learning based on Markov Decision Process (MDP) as illustrated in Figure 13. The proposal consisted of a mobile user layer, a fog layer, and a cloud layer. The algorithm applied in this study consists of three main parts: Action-selection where the system must use two methods (exploration and/or exploitation), next state observation and reward calculation and Q-network update. In this study, the delay and power consumption are formulated as follows:

$$d_{net} = \sum_{i=1}^{I} \int (k_{net}\ log_{10}\, d_i\,(t) + b_{net})\, dt \tag{9}$$

$$P_{total} = \int (\sum_{i=1}^{m} (p_{idle} + (p_{max} - p_{idle})\, x\, u_i(t))\,)\, d \tag{10}$$

This proposal aimed at reducing the cost of the power consumption and delay. The goal is to find the best strategy that minimizes C, where C is

$$min\, C = w_1 d_{total} + w_2 p_{total} + m_{total} \tag{11}$$

and m_{total} is the migration cost.

In a reinforcement learning algorithm, at each time slot or episode T_t, the agent monitors the environment and gets the state S_t then takes the action A_t according to a pre-defined strategy to maximize the reward R_{t-1}. In this study, a deep Q-learning algorithm was adopted to enable fast computation. As the problem is complex, a multi-dimensional and large scale MDP based model is designed. Perera *et al*. [38] adopted deep learning and especially deep Q-learning where it does not require prior knowledge about the whole environment. In addition, Q-learning has the advantage of fast decision-making while traditional algorithms cannot solve large-scale MDP problems with large state set and action set. Deep reinforcement learning can handle such large size problems. Similarly, study [40] proposed deep Q-network for task migration in a mobile edge computing system to generalize the experience of the agent rather than knowing all the situations which is not possible in the case of fog computing. The idea is that, at the initial state, the agent does not know how to take the action but in later states, the agent will learn which action will increase the reward. In this system, the agent is the Fog Master Controller (FMC) which has all information about the network and the servers. The state at time slot $t \in \mathcal{T}$ is defined as $s_t = u_t$ where u_t is the difference between the user's current location and the location after the movement. Hence, mobility is the main trigger for migration in this study. The FMC takes an action $a_t \in \mathcal{A}$ to migrate or not to migrate based on the state. FMC gets a reward based on its action and the current state. The reward in this study is:

$$r_t(s_t, a_t) = q(s_t) - c_t(s_t, a_t) = \begin{cases} q(s_t) & if\ a_t = a^0 \\ D - C(s_t) & if\ a_t = a^1 \end{cases} \tag{12}$$

where $q(s_t)$ is the quality of the service, D is the maximum quality $C(s_t)$ is the function of time delay.

The experiments showed that this proposal outperforms the existing approaches which use dynamic programming and the situations when "no migration" decision is taken in terms of total reward.

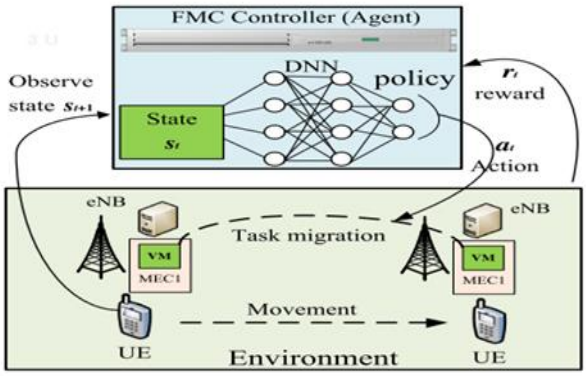

Figure 13. Deep Q-network model [39].

5.2. Framework Modeling-based Solutions

5.2.1. Follow Me Model

Study [35] proposed a framework to support smooth handover between the fog nodes in a timely manner. In this study, Mukherjee *et al*. [35] followed the principle of "follow me" like "follow me cloud" which aims at a smooth migration between one data center to another [26]. Further, there are few works using "follow me edge" or "move with me" concepts where the main trigger of migration is the mobility of the users. This is helpful in situations where mobility is the main factor in migrating. It pre-migrates the jobs when the handover is expected to happen.

This proposal results in a reduction in the service interruption time and downtime. The study guarantees services continuity and reduces the latency during

handover. It proactively takes the decision of migrating to the virtual machine. However, the limitations of this proposal are that each fog node has one Software as a Service (SAAS) server which is not practical. Further, fog nodes are assumed to be connected via wired connections, which is not always the case. In addition, getting prior knowledge of users' mobility is difficult. This study only migrates processed jobs, which is not suitable for applications with heavy data loads after job processing. However, this method is good in scenarios where user movement is fixed. This proposal developed a framework which measures the strength of the signal in the access point and then decides whether to migrate or not. This proposal results in a reduction in the service interruption time and downtime. The study guarantees services continuity and reduces the latency during handover. It proactively takes the decision of migrating to the virtual machine.

Looking at the assumptions made to design the framework, the design of the model considers access points only as fog nodes. However, the fog layer is a heterogeneous layer which consists of different types of nodes with varying hardware capabilities, architectures, OS, configuration, etc. This has been assumed across the literature [9, 13, 19, 20, 41, 42, 43, 44, 45].

5.2.2. Discovery and Deployment Model

The authors Tang *et al*. in [46], proposed a Foglets programming model that facilitates distributed programming across the resource continuum from the sensors to the cloud. In this model, the fog is augmented with the right distributed programming model. The Foglets model supports four main functionalities. Firstly, it automatically discovers fog computing resources at different levels of the network hierarchy and deploys application components onto the fog computing resources commensurate with the latency requirements of each component in the application. Secondly, it supports multi-application collate on any compute node. Thirdly, it provides communication APIs for components of the application that are deployed at different physical levels of the network hierarchy to communicate with one another to exchange application state. Lastly, it supports both latency- and workload-driven resource adaptation and state migration over space (geo- graphic) and time to deal with the dynamism in situation awareness applications.

5.2.3. Complexity Bandwidth Management Model

In Tay *et al*. [47] proposed a Settable Complexity Bandwidth Manager (SCBM) for the live migration of VMs over 5G Fog Radio Area Network (FOGRAN) Multipath Transmission Control Protocol (MPTCP) connections as illustrated in Figure 14. The study proposes SCBM for minimizing the energy consumed by wireless devices in the fog environment to sustain the migration process under different constraints on migration time and downtime. The proposal aims to optimize the energy consumption of several MPTCP connections. The migration module of this proposal focuses on how to manage the migration. The idea is to update Q value out of I_{max} rates of the pre-copy rounds that are spaced apart by $S \triangleq \frac{I_{max}}{Q}$ rounds over the round index set=$\{1,2,\ldots\ldots,I_{max}\}$ where I_{max} is the maximum number of migration pre-copy rounds, Q is the integer-valued number of the pre-copy migration rates and $S \triangleq$ is the resulting integer-valued size.

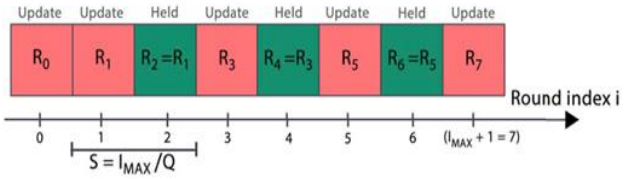

Figure 14. The main idea of proposed SCBM [46].

This study assumed that all devices are homogenous and communicate only via wireless access points. It discusses how to redirect the connection before the migration happens. It takes decisions of migration based on nearby servers and not any other parameters.

5.2.4. Multipath Transmission Control Protocol Model

Authors in Teka *et al*. [48] applied a MPTCP approach to improve the virtual machine migration time and the network transparency of the applications. The study assumes that each server in the edge/cloud environment has at least two Network Interface Cards (NICs). The migration technique followed uses MPTCP between the sender and receiver to migrate the memory and disk states in parallel, which in turn reduces the latency.

The main issue with this proposal is that it is only concerned with increasing the bandwidth so that parallel transfer is done. It is not concerned with taking decisions regarding when and where to migrate. It assumes that the decision is already taken and discusses how to migrate using multi paths to increase the bandwidth and reduce the latency. This approach is not suitable for the fog environment as fog is a dynamic environment and not implemented using servers only. Hence it is difficult to ensure that each device has at least two NICs.

Authors Kapil *et al*. in [29] suggested a nearby VM-based approach to minimize the latency and save energy. They are solving two problems. The first problem is that the mobile device's IP address is changed when the user moves from one cloudlet to another. This will terminate the established TCP connection with the virtual machine. The second problem happens when the destination cloudlet tries to access the source cloudlet during the migration.

The idea of this proposal is avoiding re-establishment

of the TCP connection by using MP TCP and ensure that one connection is opened even if the user is moving. This approach only considers re-directing the connection.

5.2.5. Layered Framework Model

Authors Teka *et al*. in [49] presented a layered approach to migrate the services from the VMs or containers. The framework is designed using an incremental file synchronization approach. This framework applies ready technologies. It consists of three layers: A base layer, which is represented by a base package with no application installed; an application layer, which contains an idle version of the application; and an instance layer, which contains the running state of the service.

5.3.Classification Based on "Why to Migrate."

Different studies have suggested different proposals to enable live VM migration to optimize one or multiple performance metrics. They applied their proposals in different scenarios and use cases. First, an analysis must be why the migration is needed. The following details the main reasons that trigger live VM migration:

- Reduced Delay: The authors Kapil *et al*. in [29] and Tay *et al*. [47] formulated the need of virtual machine migration as a way to eliminate and reduce the delay caused by service initiation time when migrating the network connection from the source to the destination in the migration process.
- Mobility: Both Mukherjee *et al*. [35] and Mouradian *et al*. [34] agreed that VM migration is triggered by the user mobility and the path should be ready for migration based on a pre-defined mobility model. Similarly, [50] agreed that mobility of the users should be studied and a proper live VM migration algorithm should be developed to follow the user mobility in order to optimize the latency incurred by this mobility. However, both Mouradian *et al*. [34] and Wang *et al*. [50] take decisions on migration proactively by following the user movement based on a mobility prediction model. [40, 48, 49, 51] all believed that mobility is the main reason to migrate the VMs according to the user movement. However, study [48] focuses on the network connection migration and the issues associated with the user mobility.
- Load Balancing: Mann in [33] stated that load balancing is crucial in fog computing as the resources are limited and hence suggested a heuristic model for live VM migration in order to optimize the load of the fog devices.
- Nodes Consolidation: The under loaded and underutilized nodes can be switched off to save energy and the VMs could be lively migrated to another fog node [39, 52].

Apart from the classification based on why to migrate, and as stated earlier, each proposal or mechanism is developed to optimize at least one performance metric. Performance metrics are the key factor in the solution design. Here, a classification of the proposals based on the purpose of migration and the performance metrics to optimize as well as the different factors that affect those metrics is provided. Table 3 discusses the different performance metrics.

Table 3. Summary of existing live VM migration solutions.

AG	L/N	M	H	FD			COF		RTA	MT		
				F	MF	T	W	WS		M	S	N
NA [50]	L	√	x	√	x	S	√		AR			√
DDP [46]	L	√	x	√	x	S	√		ITS	√		
DRL [39]	L	x	x	√	x	S	-	-	-	√		
SCBM [47]	L	√	x	√	x	S	-	-	-	√		-
BFD [33]	L	x	x	√	x	S	-	-	-	-	-	-
ILP [34]	L	√	x	√	x	S	√	√	VS	-	-	
Deep learning [53]	L	√	x	√	x	S	√		VO	√		
Generic fat tree architecture [35]	L	√	x	√	x	S	√	√	-	-	-	-
Auction algorithms [36]	-	√	x	√	x	S		√	DA	-	-	-
MPTCP [48]	L	√	x	√	x	S		√	-	√	√	√
DRL [40]	L	√	x	√	x	S		√	FR	-	-	-
PL-Edge [51]	L	√	x	√	x	S		√	VO	-	-	-
LF [49]	L	√	x	√	x	S		√	OG	-	√	√
MPTCP [29]	L	√	x	√	x	S			VS			√
SESA [37]	L	√		√	x	S	√		-	√	√	√
SMRC [38]	L	√		√	x	S		√	-			√

- The abbreviation list: AG: Algorithm used, NA: Not Applicable. L/N: Live, Non-Live Migration. M: Mobility of End Users. H: Heterogeneity of Fog nodes. FD: Fog Device. T: Type, F: Fixed, MF: Mobile. S: Same. D: Different. COF: Connectivity of Fog Devices. W: Wired. WS: Wireless. RTA: Real Time Applications.AR: Augmented Reality, VR: Virtual Reality, HL: Health, ITS: Intelligent Transportation System. FR: Face Recognition, VO: Voice Recognition, OG: Online Gaming, VS: Vehicular System, DA: Divisible applications, MT: Migration Type, M: Memory, S: Storage, N: Network. BFD: Best Fit Decreasing. DDP: Discovery and Deployment Protocol. DRL: Deep Reinforcement Learning algorithms. SCBM: Settable-Complexity Bandwidth Manager, TCBM: Tunable-Complexity Bandwidth Manager. PS: Policies and Strategies, MPTCP: Multipath TCP, PL-Edge: Policy-VM Latency-aware consolidation scheme for mobile Edge computing, LF: Layered Framework.

From Table 3, we can see that most of the studies proposed solutions to optimize more than one performance metric simultaneously, while few others focused on one metric only. Different performance metrics are studied in the literature and explored by us including downtime, TMT, mean time to repair, mean

time to recover, failure rate, latency, energy saving, bandwidth utilization and resource utilization. For each performance metric, there are factors that hinder the achievement of the best values. The following factors were identified affecting the performance metrics: Network bandwidth, heterogeneity of the fog devices, mobility of the users and fog devices, late decision to migrate the VMs and early handover for the VMs, and data size.

Further, it is noticed that the two most common metrics are latency and downtime optimization. Authors in Mouradian *et al*. [34], Teka *et al*. [48], and Wang *et al*. [50] proposed solutions to reduce the latency and downtime caused by live VM migration. They stated that the main purpose of live virtual machine migration is to transfer the tasks of the user without interrupting the service, especially for time-sensitive applications. Hence, their solutions are developed in such a way to reduce the latency and downtime. However, those metrics are affected by many factors. Mobility of the devices is the main factor. When users or fog devices move, the downtime is affected. Another factor with impact is the network bandwidth [15]. When the bandwidth is low, the latency and downtime are high. [31] used Myifogsim and [50] used prototype testing.

The author Mann in [33] stated that load balancing is one of the key factors to help fog computing achieving better resource utilization as fog devices are resource-constrained devices. The author Mann [33] live VM migration as a powerful tool for resource management, so it is crucial to balance the load in the fog layer to avoid any overloading or underloading. They used Cloudsim to test the load balancing metric. However, this metric is affected by the network bandwidth and heterogeneity of the fog devices. Therefore, it becomes difficult to balance the load across different devices with different computational resources and functionalities (e.g., routers, access points, base stations, vehicles, etc.,). Both Naha *et al*. [36] and Tay *et al*. [47] agreed that the smart and edge devices suffer from limited resources and energy. They stated that energy consumption still offsets the benefits of live VM migration. Therefore, there is a need to optimize energy consumption to help the edge and mobile devices to meet the growing needs of low-latency and resource-intensive applications. However, as with any other performance metrics, energy consumption is affected by several factors including the size of the data to be migrated and the mobility of the users.

To conclude, it is noted that no study addresses the availability and robustness of the migrated VMs. Further, no study discussed how to optimize the Mean Time to Repair and the Mean Time to Recover (MTTR and MTTRe) especially when a wrong decision is taken and when the migration fails. These metrics are affected by many factors including the heterogeneity of fog devices and mobility.

Both MTTR and MTTRe can be defined as follows:

- Mean Time To Repair (MTTR): it is the amount of time required to repair the failure of live VM Migration on a wrong decision and restore the service to functionality on an appropriate fog node.

$$MTTR = \frac{total\ repair\ time}{total\ number\ of\ repairs} \tag{13}$$

- Mean Time To Recover (MTTR): it is the average time from the time the live VM migration fails until the time it is resumed on another fog node successfully.

$$MTTRe = \frac{Total\ period\ of\ downtime\ (\ out\ of\ service)}{total\ number\ of\ failures} \tag{14}$$

Looking at other comparison factors and when discussing live virtual machine migration, a few more criteria need to be considered. The following discusses some of those criteria:

- Granularity: this is concerned with whether the proposed solutions consider multiple migrations at the same time or a single migration only. Most of the studies in the literature mentioned multiple migrations they did not reflect that in the designs of their solutions.
- Migration decision time: this criterion is concerned with the timing of the migration. It is worth noting that most of the studies proposed reactive designs in which VM migration is triggered by some factors like overloading of the physical machine and user mobility. This reactive decision making may lead to late handover. The migration will be done after the trigger is detected. It then takes time to decide and choose an appropriate fog device to host the virtual machine and then migrate.

However, a few studies only proactively migrate VMs based on different inputs. For example, the study in [34] takes the mobility prediction of the user from GPS input and proactively decides on when and where to migrate. This is better than a reactive solution as, according to the study, it can decrease the latency and the number of migrations needed. However, this also may incur some issues like early handover and reduced reliability as it should not depend only on the mobility prediction as fog environment has some other several characteristics which might affect the decision such as: Decentralization nature, heterogeneity, location-awareness, etc. Such solutions should consider the issue of the user changing movement pattern.

- Decision to be taken: Each study takes a decision related to the time to migrate or the place to migrate to. Most of the studies take decisions regarding where to migrate. Only two studies discussed when to migrate. Study [50] discusses when to migrate by pre-migrating the jobs to ensure that the migration decision is taken and the process is started before the full handover. However, the authors in Mouradian *et al*. [34], addressed when and where to migrate.

Regarding "when to migrate," they refer to ready mobility models to take the decision in a pro-active manner. As for "when to migrate," it is based on the resource's availability. It is the only study in the literature we came across which considers both decisions.

5.4. Studies on "What Component to Migrate"

Here a classification about the studies based on what element is migrated is discussed. It can be a service, a task, a job, the VM itself, or an application component.

- Task: It is an execution of a single process in the fog node.
- Job: It is a collection of multiple tasks on the fog node.
- Service: It is software that performs tasks. It automatically starts in the background when the tasks start.
- Application: It is a program which interacts with the user to accomplish a particular task.
- Virtual Machine: It is a system image that behaves like an actual computer and shares the physical computer resources including memory, CPU, and disk resources.

It is noted that the most used terms in the literature are the service and the task while in the implementation, the full virtual machine migration is considered. There is another classification based on what to be migrated which considers the environment to be migrated whether it is a VM or a container. It is worth noting that most of the studies focus on virtual machine migration although some recent studies considered container migration because containers are lighter than VM and require less time to boot and initialize in contrast of the virtual machine which needs more time to boot and results in a delay. Table 4 summarizes the studies based on what is migrated.

Table 4. Migrated components.

Migrated component	Reference	Use case
Task	[40][36]	FR, vehicles
Service	[33] [49]	Video surveillance, OG, face detection
Application	[46]	Vehicles
Job	[50]	Smart sport and smart tourism
Full VM	[28], [34], [35], [47], [48], [51], [53]	Vehicles, VO applications

In this classification, we differentiate between the studies based on the components they migrate. As shown in Table 5 most of the studies across in the literature address the migration of the virtual machine itself. Some studies, however, do address the migration of services and tasks while others address the migration of jobs and applications.

Table 5. What to migrate.

Environment	VM	Container
Literature studies	[29], [33], [34], [35], [36], [40], [47], [48], [49], [50] , [51], [53], [54]	[39], [46], [49]

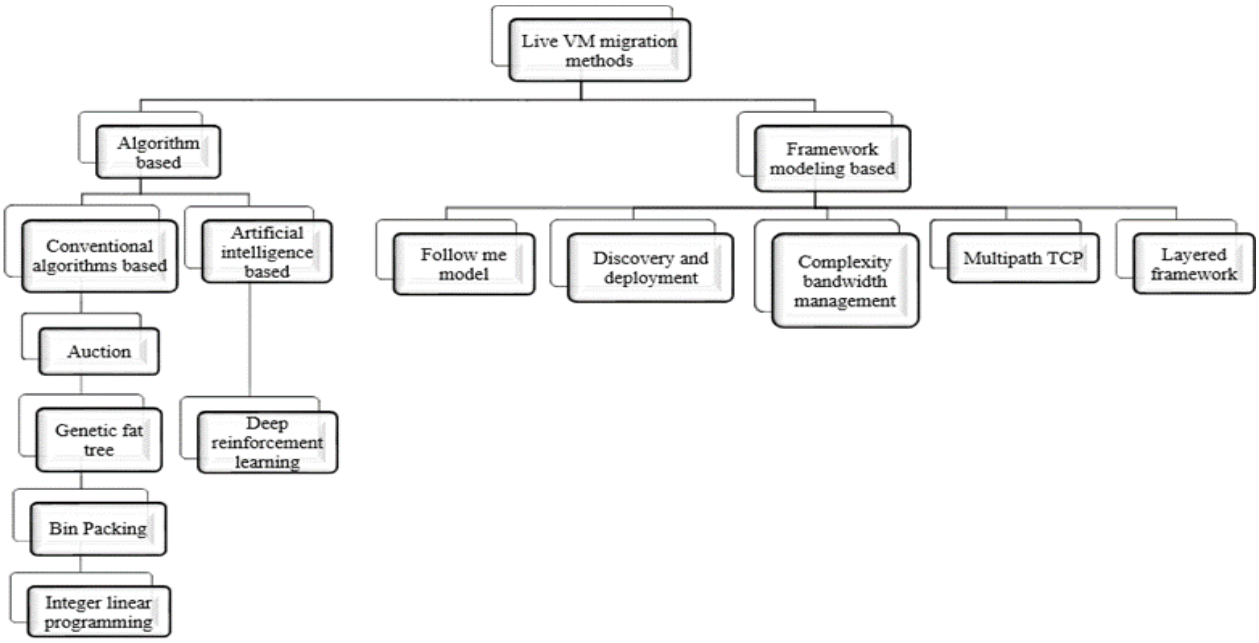

Figure 15. Classification of live VM migration methods in fog computing.

To link the components to migration, the environment of migration is studied whether it is a virtual machine migration or a container migration. It is noticed that most of the studies fall under the virtual machine migration category although some recent studies moved towards container migration. Table 6 concludes the classification with the factors regarding when and where to migrate.

Table 6. Common classification factors.

Study	Granularity (S/M)		Timing of the decision		Nature of decision	
	Single	Multiple	Reactive	Proactive	When	Where
[51]	√			√	√	
[47]		√	√	√		√
[40]	√		√			√
[48]		√	√			√
[34]	√		√			√
[35]	√			√	√	√
[54]	√	√			√	√
[36]		√	√			√
[37]		√	√			√
[49]		√	√		-	-
[50]	√		√		-	-
[41]		√		√	√	
[52]	√		√			√

Table 7 details the ket findings of exisitng models of VM migration.

Table 7. Key findings of existing models of VM migration.

Evaluation and analysis of existing live VM migration solutions.	Key concepts/ Findings
	• Two categories of algorithm-based solutions. Traditional/conventional algorithms and AI-based algorithms. • AI-based solutions are more suitable for solving complex and specialized • A traditional algorithm solution is more suitable for broad problems. • Algorithms can be classified based on: "why to migrate," "what component to migrate."

6. Container versus Virtual Machine

When discussing VM migration, it must not ignore the discussion about the containers. Although there are many common points between them, there are also many differences worth discussing [35]. Containerization is an OS-Level virtualization where the kernel is shared between the users. Multiple users can use the kernel's resources simultaneously in an isolated environment. The virtualization instances in that case are called containers [13]. The software in the container can share the resources provided by the kernel and assigned to this container [13].

To compare between container and virtual machine, containers are OS-Level virtualization while VM implies both OS-level and hardware-level virtualization. Containers are lighter in weight than VM which might result in less challenges than VM [40]. According to [55], the container has several advantages over VMs. However, there are a few differences like the performance, size, and ease of use.

Containers require less time to boot and initialize in contrast to the virtual machine which needs more time to boot which in turn leads to a delay in the process. Hence, container is faster than VM [56]. Recalling how a virtual machine migrates, the memory state is copied and transferred to the destination over several iterations until all dirty pages are transferred. Meanwhile, the storage is transferred, and the network connection is redirected. The iterations take longer time if the iterations take longer time if the memory pages are big [56].

On the other hand, containers are lighter, and the process of migration them is different. It does not incur a large amount of time. The process of the migration is done by simply killing the container at the source and recreating it on the destination physical host. This process is known as Checkpoint and Restore where a checkpoint is created, and the state of the current applications is restored at the destination.

Although container has several advantages over virtual machine, but it also has several issues not suffered by VM. For example, as containers are sharing the kernel resources, they also share some libraries. This means that during the migration process, the destination should ensure that those libraries are prepared and ready before the migration [35]. In contrast, VM does not require preparing the libraries before migration.

Table 8 illustrates the key points of the comparison between VM and containers.

Table 8. Key findings of containers Vs.VM.

Container Versus Virtual Machine	Key Concepts/ Findings
	• Containerization is an OS-level virtualization • VM implies both OS-level and hardware-level virtualization. • Containers are lighter in weight than VM containers require less time to boot and initialize in contrast to the virtual machine which needs more time.

Table 9 lists the summary points of open questions and challenges.

Table 9. Summary of open questions and challenges.

Open questions and challenges.	Key concepts/ Findings
	• Heterogeneity of the fog devices. • Priority of fog devices. • When and where to migrate. • Migration overheads. • Mobility models.

7. Open Questions and Challenges

To summarize, all the studies have some common research gaps and remaining research questions to address including the following:

- Heterogeneity of the fog devices: According to Bittencourt *et al*. [13], fog computing is characterized by the heterogeneity of nodes and hence there is a need for virtualization mechanisms that run on different types of fog nodes. As seen across the literature, most of the studies ignore this aspect and assume that fog devices are homogenous. So, how will the heterogeneous devices handle the migration without affecting the running services and with minimum downtime?.
- Priority of fog devices: As stated in Habibi *et al*. [25], each fog is responsible to do its own functions and the fog-related functions always come in the second priority. This would prevent the real-time applications from achieving their objective of low latency and will incur delays. Hence, a remaining question is how to balance between the devices' own

functions and the fog-related functions to optimize the performance of real-time applications.

- When and where to migrate: The two aspects should be studied together and not independently. Most of the studies design their solutions to decide where to migrate without paying attention to when to migrate. This might cause a problem of late or early handover and hence failure of the migration process.
- Migration overheads: According to Mukherjee *et al*. [35], a good VM live migration strategy should consider in its design the overhead incurred by the migration and try to minimize it. Such overheads include computational overhead, network overhead and space overhead. Most of the existing proposals did not consider the overhead reduction in their proposals.
- Mobility models: they should be more accurate by considering the possible changes in the expected user mobility pattern. Most of the studies used ready mobility models which assume unrealistically that the user movement is fixed, and the user moves in one specific direction. The design should consider possible changes in the expected user movement pattern.
- Speed: Fog network is closer to the end users which enables the users to run real-time applications as it will run on fog layer which will be faster [57].

8. Conclusions and Future Work

To conclude, live VM migration is a key technique for real-time applications, but it has several challenges that need addressing. This paper summarizes the state-of-the-art technologies in live virtual machine migration in the field of fog computing. All the technologies are compared with each other by using the performance metrics and overheads discussed from the review, the following conclusions were drawn:

1) Most virtual machine algorithms decide only to check either when to migrate the virtual machine or where to migrate. Few studies considered both factors in the decision.
2) Most of the studies ignored the fact that the fog nodes are heterogenous and assumed that the node are homogenous.
3) Most of the studies did not consider the issue of early and late handover although it is a critical factor in real-time applications run on fog layer.

The paper also presented a new classification for the existing literature. It divided the solutions based on algorithms based and framework model based and further broken into subcategories, where algorithm-based classification is divided into conventional based and AI-based.

References

[1] Aazam M., Zeadally S., and Harras K., "Offloading in Fog Computing for IoT: Review, Enabling Technologies, and Research Opportunities," *Future Generation Computer Systems*, vol. 87, pp. 278-289, 2018. https://doi.org/10.1016/j.future.2018.04.057

[2] Abou-Tair D., Büchsenstein S., and Khalifeh A., "A Fog Computing-Based Framework for Privacy Preserving IoT Environments," *The International Arab Journal of Information Technology*, vol. 17, no. 3, pp. 306-315, 2020. https://doi.org/10.34028/iajit/17/3/4

[3] Ahmed A., Arkian H., Battulga D., Fahs A., and Farhadi M., "Fog Computing Applications: Taxonomy and Requirements," *arXiv Preprint*, arXiv:1907.11621, 2019. https://doi.org/10.48550/arXiv.1907.11621

[4] Ahmad K., Mohammad O., Atieh M., and Ramadan H., "Enhanced Performance and Faster Response using New IoT Litetechnique," *The International Arab Journal of Information Technology*, vol. 16, no. 3A, pp. 548-556, 2019. https://iajit.org/PDF/Special%20Issue%202019, %20No.%203A/18601.pdf

[5] Akintoye S. and Bagula A., "Improving Quality-of-Service in Cloud/Fog Computing through Efficient Resource Allocation," *Sensors*, vol. 19, no. 6, pp. 1-29, 2019. https://doi.org/10.3390/s19061267

[6] Amendola D., Cordeschi N., and Baccarelli E., "Bandwidth Management VMs Live Migration in Wireless Fog Computing for 5G Networks," *in Proceedings of the 5th IEEE International Conference on Cloud Networking (Cloudnet)*, Pisa, pp. 21-26, 2016. DOI:10.1109/CloudNet.2016.36

[7] Anawar M., Wang S., Zia M., Jadoon A., Akram U., and Raza S., "Fog Computing: An Overview of Big IoT Data Analytics," *Wireless Communications and Mobile Computing*, vol. 2018, pp. 1-23, 2018. https://doi.org/10.1155/2018/7157192d

[8] Atlam H., Walters R., and Wills G., "Fog Computing and the Internet of Things: A Review," *Big Data and Cognitive Computing*, vol. 2, no. 2, pp. 1-18, 2018. https://doi.org/10.3390/bdcc2020010

[9] Baccarelli E., Scarpiniti M., and Momenzadeh A., "Fog-Supported Delay-Constrained Energy-Saving Live Migration of VMs over MultiPath TCP/IP 5G Connections," *IEEE Access*, vol. 6, pp. 42327-42354, 2018. DOI:10.1109/ACCESS.2018.2860249

[10] Bao W., Yuan D., Yang Z., Wang S., Li W., Zhou B., and Zomaya A., "Follow Me Fog: Toward Seamless Handover Timing Schemes in a Fog

Computing Environment," *IEEE Communications Magazine*, vol. 55, no. 11, pp. 72-78, 2017. DOI:10.1109/MCOM.2017.1700363

[11] Besharati R. and Rezvani M., "A Prototype Auction-based Mechanism for Computation Offloading in Fog-Cloud Environments," *in Proceedings of the 5th Conference on Knowledge Based Engineering and Innovation*, Tehran, pp. 542-547, 2019. DOI:10.1109/KBEI.2019.8734918

[12] Bi Y., Han G., Lin C., Deng Q., Guo L., and Li F., "Mobility Support for Fog Computing: An SDN Approach," *IEEE Communications Magazine*, vol. 56, no. 5, pp. 53-59, 2018. DOI:10.1109/MCOM.2018.1700908

[13] Bittencourt L., Immich R., Sakellariou R., Fonseca N., and Madeira E., "The Internet of Things, Fog and Cloud Continuum: Integration and Challenges," *Internet of Things*, vol. 2-3, pp. 134-155, 2018. https://doi.org/10.1016/j.iot.2018.09.005

[14] Bittencourt L., Lopes M., Petri I., and Rana O., "Towards Virtual Machine Migration in Fog Computing," *in Proceedings of the 10th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing*, Krakow, pp. 1-8, 2015, DOI:10.1109/3PGCIC.2015.85

[15] Botta A., De Donato W., Persico V., and Pescapé A., "Integration of Cloud Computing and Internet of Things: A Survey," *Future Generation Computer Systems*, vol. 56, pp. 684-700, 2016. https://doi.org/10.1016/j.future.2015.09.021

[16] Chaufournier L., Sharma P., Le F., Nahum E., Shenoy P., and Towsley D., "Fast Transparent Virtual Machine Migration in Distributed Edge Clouds," *in Proceedings of the 2nd ACM/IEEE Symposium on Edge Computing*, Wilmington, pp. 1-13, 2017. https://doi.org/10.1145/3132211.3134445

[17] Choudhary A., Govil M., Singh G., Awasthi L., Pilli E., and Kapil D., "A Critical Survey of Live Virtual Machine Migration Techniques," *Journal of Cloud Computing: Advances, Systems and Applications*, vol. 6, no. 1, pp. 1-41, 2017. DOI:10.1186/s13677-017-0092-1

[18] Damania K., Holmukhe S., Singhai V., and Bhavathankar P., "An Overview of VM Live Migration Strategies and Technologies," *in Proceedings of the 2nd International Conference on Electronics, Communication and Aerospace Technology*, Coimbatore, pp. 1185-1190, 2018. DOI:10.1109/ICECA.2018.8474910

[19] Doan T., Nguyen G., Salah H., Pandi S., Jarschel M., Pries R., and Fitzek F., "Containers vs Virtual Machines: Choosing the Right Virtualization Technology for Mobile Edge Cloud," *in Proceedings of the IEEE 2nd 5G World Forum*, Dresden, pp. 46-52, 2019. DOI: 10.1109/5GWF.2019.8911715

[20] Duan J., Ren K., Zhou W., Xu Y., and Dou W., "A Service Migration Method for Resource Competition in Mobile Edge Computing," *in Proceedings of the IEEE International Performance, Computing, and Communications Conference*, Austin, pp. 1-8, 2021. DOI:10.1109/IPCCC51483.2021.9679421

[21] Genez T., Tso F., and Cui L., "Latency-Aware Joint Virtual Machine and Policy Consolidation for Mobile Edge Computing," *in Proceedings of the 15th IEEE Annual Consumer Communications and Networking Conference*, Las Vegas, pp. 1-6, 2018. DOI:10.1109/CCNC.2018.8319204

[22] Giri A., Dutta S., Neogy S., Dahal K., and Pervez Z., "Internet of things (IoT): A Survey on Architecture, Enabling Technologies, Applications and Challenges," *in Proceedings of the 1st International Conference on Internet of Things and Machine Learning IML'17*, Liverpool, pp. 1-12, 2017. https://doi.org/10.1145/3109761.3109768

[23] Goncalves D., Velasquez K., Curado M., Bittencourt L., and Madeira E., "Proactive Virtual Machine Migration in Fog Environments," *in Proceedings of the IEEE Symposium on Computers and Communications*, Natal, pp. 00742-00745, 2018. DOI:10.1109/ISCC.2018.8538655

[24] Govindaraj K. and Artemenko A., "Container Live Migration for Latency Critical Industrial Applications on Edge Computing," *in Proceedings of the IEEE 23rd International Conference on Emerging Technologies and Factory Automation*, Turin, pp. 83-90, 2018. DOI10.1109/ETFA.2018.8502659

[25] Habibi P., Farhoudi M., Kazemian S., Khorsandi S., and Leon-Garcia A., "Fog Computing: A Comprehensive Architectural Survey," *IEEE Access*, vol. 8, pp. 69105-69133, 2020. DOI:10.1109/ACCESS.2020.2983253

[26] Haouari F., Faraj R., and AlJa'am J., "Fog Computing Potentials, Applications, and Challenges," *in Proceedings of the International Conference on Computer and Applications*, Beirut, pp. 399-406, 2018. DOI:10.1109/COMAPP.2018.8460182

[27] Hu P., Dhelim S., Ning H., and Qiu T., "Survey on Fog Computing: Architecture, Key Technologies, Applications and Open Issues," *Journal of Network and Computer Applications*, vol. 98, pp. 27-42, 2017. https://doi.org/10.1016/j.jnca.2017.09.002

[28] Jiang Y., Huang Z., and Tsang D., "Challenges and Solutions in Fog Computing Orchestration," *IEEE Network*, vol. 32, no. 3, pp. 122-129, 2018. DOI:10.1109/MNET.2017.1700271

[29] Kapil D., Pilli E., and Joshi R., "Live Virtual

Machine Migration Techniques: Survey and Research Challenges," *in Proceedings of the 3rd IEEE International Advance Computing Conference*, Ghaziabad, pp. 963-969, 2013. DOI:10.1109/IAdCC.2013.6514357

[30] Kaur A., Kumar S., Gupta D., Hamid Y., Hamdi M., and KsibiA., "Algorithmic Approach to Virtual Machine Migration in Cloud Computing with Updated SESA Algorithm," *Sensors*, vol. 23, no. 13, pp. 1-18, 2023. https://doi.org/10.3390/s23136117

[31] Li S., Tryfonas T., and Li H., "The Internet of Things: A Security Point of View," *Internet Research*, vol. 26, no. 2, pp. 337-359, 2016. DOI: 10.1108/IntR-07-2014-0173

[32] Machen A., Wang S., Leung K., Ko B., and Salonidis T., "Live Service Migration in Mobile Edge Clouds," *IEEE Wireless Communications*, vol. 25, no. 1, pp. 140-147, 2018. DOI:10.1109/MWC.2017.1700011

[33] Mann Z., "Notions of Architecture in Fog Computing," *Computing*, vol. 103, no. 1, pp. 51-73, 2021. https://doi.org/10.1007/s00607-020-00848-z

[34] Mouradian C., Naboulsi D., Yangui S., Glitho R., Morrow M., and Polakos P., "A Comprehensive Survey on Fog Computing : State-of-the-Art and Research Challenges," *IEEE Communications Surveys and Tutorials*, vol. 20, no. 1, pp. 416-464, 2018. DOI: 10.1109/COMST.2017.2771153

[35] Mukherjee M., Shu L., and Wang D., "Survey of Fog Computing: Fundamental, Network Applications, and Research Challenges," *IEEE Communications Surveys and Tutorials*, vol. 20, no. 3, pp. 1826-1857, 2018. DOI:10.1109/COMST.2018.2814571

[36] Naha R., Garg S., Georgakopoulos D., Jayaraman P., Gao L., Xiang Y., and Ranjan R., "Fog Computing: Survey of Trends, Architectures, Requirements, and Research Directions," *IEEE Access*, vol. 6, pp. 47980-48009, 2018. DOI:10.1109/ACCESS.2018.2866491

[37] Noura M., Atiquzzaman M., and Gaedke M., "Interoperability in Internet of Things : Taxonomies and Open Challenges Interoperability in Internet of Things : Taxonomies and Open Challenges," *Mobile Networks and Applications*, vol. 24, pp. 796-809, 2019. https://doi.org/10.1007/s11036-018-1089-9

[38] Perera C., Qin Y., Estrella J., Reiff-Marganiec S., and Vasilakos A., "Fog Computing for Sustainable Smart Cities: A Survey," *ACM Computing Surveys*, vol. 50, no. 32, pp. 1-43, 2017. https://doi.org/10.1145/3057266

[39] Perera C., Zaslavsky A., Christen P., and Georgakopoulos D., "Context Aware Computing for the Internet of Things: A Survey," *IEEE Communications Surveys and Tutorials*, vol. 16, no. 1, pp. 414-454, 2014. DOI:10.1109/SURV.2013.042313.00197

[40] Portnoy M., *Virtualisation Essesntials*, John Wiley and Sons, 2012. https://www.gettextbooks.com/isbn/9781118176719/

[41] Puliafito C., Vallati C., Mingozzi E., Merlino G., Longo F., and Puliafito A., "Container Migration in the Fog: A Performance Evaluation," *Sensors*, vol. 19, no. 7, pp. 1-22, 2019, https://doi.org/10.3390/s19071488

[42] Roig P., Alcaraz S., Gilly K., and Juiz C., "Modelling VM Migration in a Fog Computing Environment," *Elektronika Ir Elektrotechnika*, vol. 25, no. 5, pp. 75-81, 2019. https://doi.org/10.5755/j01.eie.25.5.24360

[43] Saurez E., Hong K., Lillethun D., Ramachandran U., and Ottenwälder B., "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*, California, pp. 258-269, 2016. https://doi.org/10.1145/2933267.2933317

[44] Sheng J., Hu J., Teng X., Wang B., and Pan X., "Computation Offloading Strategy in Mobile Edge Computing," *Information*, vol. 10, no. 6, pp. 1-20, 2019. https://doi.org/10.3390/info10060191

[45] Singh G. and Gupta P., "A Review on Migration Techniques and Challenges in Live Virtual Machine Migration," *in Proceedings of the 5th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)*, Noida, pp. 542-546, 2016. DOI: 10.1109/ICRITO.2016.7785015

[46] Tang Z., Zhou X., Zhang F., Jia W., and Zhao W., "Migration Modeling and Learning Algorithms for Containers in Fog Computing," *IEEE Transactions on Services Computing*, vol. 12, no. 5, pp. 712-725, 2019. DOI:10.1109/TSC.2018.2827070

[47] Tay Y., Gaurav K., and Karkun P., "A Performance Comparison of Containers and Virtual Machines in Workload Migration Context," *in Proceedings of the IEEE 37th International Conference on Distributed Computing Systems Workshops*, Atlanta, pp. 61-66, 2017. DOI:10.1109/ICDCSW.2017.44

[48] Teka F., Lung C., and Ajila S., "Nearby Live Virtual Machine Migration Using Cloudlets and Multipath TCP," *Journal of Cloud Computing*, vol. 5, no. 1, pp. 1-21, 2016. https://doi.org/10.1186/s13677-016-0061-0

[49] Teka F., Lung C., and Ajila S., "Seamless Live Virtual Machine Migration with Cloudlets and Multipath TCP," *in Proceedings of the IEEE 39th Annual Computer Software and Applications Conference*, Taichung, pp. 607-616, 2015.

DOI: 10.1109/COMPSAC.2015.31

[50] Wang S., Xu J., Zhang N., and Liu Y., "A Survey on Service Migration in Mobile Edge Computing," *IEEE Access*, vol. 6, pp. 23511-23528, 2018. DOI:10.1109/ACCESS.2018.2828102

[51] Xu X., Liu Q., Qi L., Yuan Y., Dou W., and Liu A., "A Heuristic Virtual Machine Scheduling Method for Load Balancing in Fog-Cloud Computing," *in Proceedings of the IEEE 4th International Conference on Big Data Security on Cloud, IEEE International Conference on High Performance and Smart Computing, and IEEE International Conference on Intelligent Data and Security*, Omaha, pp. 83-88, 2018. DOI:10.1109/BDS/HPSC/IDS18.2018.00030

[52] Yi S., Hao Z., Qin Z., and Li Q., "Fog Computing: Platform and Applications," *in Proceedings of the 3rd IEEE Workshop on Hot Topics in Web Systems and Technologies*, Washington (DC), pp. 73-78, 2016. DOI: 10.1109/HotWeb.2015.22

[53] Yousefpour A., Fung C., Nguyen T., Kadiyala K., Jalali F., Charlotte U., and Jue J., "All One Needs to Know about Fog Computing and Related Edge Computing Paradigms : A Complete Survey," *Journal of Systems Architecture*, vol. 98, pp. 289-330, 2019. https://doi.org/10.1016/j.sysarc.2019.02.009

[54] Zhang C. and Zheng Z., "Task Migration for Mobile Edge Computing Using Deep Reinforcement Learning," *Future Generation Computer Systems*, vol. 96, pp. 111-118, 2019. https://doi.org/10.1016/j.future.2019.01.059

[55] Zhang F., "Challenges and New Solutions for Live Migration of Virtual Machines in Cloud Computing Environments," Doctoral Theses, Georg-August University, 2018. file:///C:/Users/user/Downloads/dissertation-submission.pdf

[56] Zhang F., Liu G., Fu X., and Yahyapour R., "A Survey on Virtual Machine Migration: Challenges, Techniques, and Open Issues," *IEEE Communications Surveys and Tutorials*, vol. 20, no. 2, pp. 1206-1243, 2018. DOI:10.1109/COMST.2018.2794881

[57] Zhou Z., Liao H., Zhao X., Ai B., and Guizani M., "Reliable Task Offloading for Vehicular Fog Computing under Information Asymmetry and Information Uncertainty," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 9, pp. 8322-8335, 2019. DOI:10.1109/TVT.2019.2926732.

Shahd Alqam received the bachelor and M.Sc. degree in Engineering in Computer hardware from Coventry University-UK, in 2011 and 2013 respectively. She is currently continuing her Ph. D study in Computer Science from Sultan Qaboos University-Oman. Her research interest is Computer hardware, Networking, Cybersecurity and Internet of Things.

Nasser Alzeidi received his PhD degree in computer science from the University of Glasgow (UK) in 2007. He is currently an Associate Professor of computer science and the Deputy Vice Chancellor for Administrative and Financial affairs at Sultan Qaboos University. His research interests include IoT architectures, Blockchain, performance evaluation of communication systems, wireless networks, interconnection networks, System on Chip architectures and parallel and distributed computing. Dr. Alzeidi supervised a large number of M.Sc. and PhD students in different Computer Science and related areas. He has been the director for the Center for Information Systems and led the digital transformation projects and initiatives at Sultan Qaboos University for seven years. He is a member of editorial boards of international journals and conference committees. He has developed a number of cutting-edge courses and given many keynote talks, conference presentations and media interviews. He is also a member of the IEEE.

Abderezak Touzene is a full professor and Head of Department Computer Science at the College of Science, Sultan Qaboos University, Muscat, Sultanate of Oman. He has more than 30 years of teaching and research experience. He obtained his Ph.D. from Institute Polytechnique de Grenoble, France (1992), M.Sc. from Paris University, France (1989), and B.Sc. from University of Technology Houari Boumediene, Algeria (1987) His area of interest includes Smart Systems, Cloud computing, Cybersecurity, Machine Learning, Big Data analytics, IoT, etc. In the last five years, he has published 23 Journals and 11 conference publications. He has received 8 research grants and is on the editorial board of several journals and conferences.

Khaled Day received the MSc and PhD degrees in computer science from the University of Minnesota, USA, in 1989 and 1992 respectively. He is currently holding a professor position at the Department of Computer Science in Sultan Qaboos University, Oman. His research interests include parallel and distributed computing and networks.