Fog and Cloud Computing Optimization in Mobile IoT Environments

José Carlos Ribeiro Vieira
josecarlosvieira@tecnico.ulisboa.pt

Instituto Superior Técnico, Universidade de Lisboa
Advisors: Prof. António Manuel Raminhos Cordeiro Grilo
Prof. João Coelho Garcia

Abstract. We introduce a xxxxx

Keywords: Cloud computing, fog computing, mobility, optimization, multi-objective

Table of Contents

1 Introduction

World is growing at a fast pace and so is data. Agility and flexibility of big data applications are gradually taking the form of the Internet of Things (IoT) and there's no doubt that it is a great resource. It comprises *things* that have unique identities and are connected to the Internet. Ubiquitous deployment of interconnected devices is estimated to reach 50 billion units by 2020 [1]. This exponential se is broadly supported by the increasing number of mobile devices (e.g., smart phones, tablets), smart sensors (e.g., autonomous transportation, industrial controls, wearables), wireless sensors and actuators networks. The number of mobile devices are predicted to reach 11.6 billion by 2021, exceeding the world's projected population at that time (7.8 billion), where the subset of wearable ones are expected to be 929 million [2].

Managing the data generated by IoT sensors and actuators is one of the biggest challenges faced when deploying an IoT system. Although this kind of devices has evolved radically in the last years, battery life, computation and storage capacity remain limited. This means that they are not suitable for running heavy applications, being necessary, in this case, to resort to third parties.

Cloud computing (CC) is a resource-rich environment that has been imperative in expanding the reach and capabilities of IoT devices. It enables that clients outsource the allocation and management of resources (hardware or software) that they rely upon to the cloud. In addition, to avoid over- or under-provisioning, cloud service providers (CSPs) also afford dynamic resources for a scalable workload, applying a pay-as-you-go cost model. This way, cloud computing is the on-demand delivery of compute power, database storage, applications, and other IT resources through a cloud services platform via the internet with pay-as-you-go pricing. [3]. CC, besides overcoming the aforementioned limitations, also brings other advantages such as availability, flexibility, scalability, reliability, to mention a few.

Despite the benefits of using cloud computing, two main problems, linked to IoT applications, remain unresolved, and they cannot be underestimated. The first and the most obvious, is the fact that cloud servers reside in remote data center. Consequently, the end-to-end communication have long delays (characteristic of multi-hops transmissions over the Internet). Some applications, with ultra-low latency requirements, can't support such delays. Augmented reality applications that use head-tracked systems, for example, require end-to-end latencies to be less than 16 ms [4]. Cloud-based virtual desktop applications require end-to-end latency below 60 ms if they are to match QoS of local execution [5]. Remotely rendered video conference, on the other hand, demand end-to-end latency below 150 ms [6]. On the other hand, the exponential growing number of IoT devices raises the other problem. The basic premise is that as the number of connected devices increases, the bandwidth required to support them becomes too large for centralized processing (i.e. CC). As a fallout of this sense-process-actuate model, where the processing is done in the cloud, there are two immediately apparent options: (1) increase the number of centralized cloud data centers, what will be too costly, and (2) get more efficient with the data sent to the cloud.

To overcome these limitations, the solution that has already been proposed is to bring the cloud closer to the end devices, where entities such as base-stations would host smaller sized clouds. This idea has brought the emergence of several computing paradigms such as cloudlets [7], fog computing [8], edge computing [9], and follow me cloud [10], to name a few. These are different solutions, often confused in the literature, that provide faster approaches

and gain better situational awareness in a far more timely manner. However, they all share the same goal, achieve the option number (2), doing more processing in the cloud-to-things continuum instead of in the cloud.

As it will be discussed later, fog computing, also known as fog networking or fogging, is the most comprehensive and natural paradigm to get more efficient with the data sent to the cloud. A simple definition of fog is "cloud closer to the ground", which gives an idea of its functioning. It is a decentralized computing infrastructure that aims to enable computing, storage, networking, and data management not only in the cloud, but also along the cloud-to-thing path as data traverses to the cloud. Essentially it extends cloud computing and services along the network itself, bringing them closer to where data is created and acted upon. Fog brought these services closer to the end devices due to its low hardware footprint and low power consumption. This way, both problems raised by the use of cloud computing, are solved. First, the path travelled by the data in the sense-process-actuate architecture is much smaller (ideally, just one hop to send data and another to receive the results), allowing latency to be much smaller compared to the traditional cloud computing. Second, through geographical distribution, there is a significant amount of data that are no longer travelling to the cloud. Fog computing, prefers to process data, as much as possible, in the nodes closer to the edge. In a simplistic manner, it only considers transferring the data further if there is not enough computational power to meet the demands.

Nevertheless, cloud is still more suitable than fog for massive data processing, when the latency constraints are not so tight. Therefore, even though fog computing has been proposed to grant support for IoT applications, it does not replace the needs of cloud-based services. In fact, fog and cloud complement each other, and one cannot replace the need of the other. Together, they offer services even further optimized to IoT applications. Moreover, Internet connectivity is not essential for the fog-based services to work, what means that services can work independently and send necessary updates to the cloud whenever the connection is available [11].

Nonetheless, it is worth mentioning that similarly to cloud computing, fog can use the concept of virtualization to grant heterogeneity. IoT applications may span many operating systems and application environments (e.g., Android, iOS, Linux, Windows), as well as diverse approaches to partitioning and offloading computation. There is churn in this space from new OS versions, patches to existing OS versions, new libraries, new language run-time systems, and so on. In order to fog support all these variants, it can be introduced a level of abstraction that cleanly encapsulate this messy complexity in a virtual machine (VM). Also, it enables VMs to coexist in a physical server (host) to share resources. Meanwhile, in fog computing, the use of VMs is also crucial to provide mobility. As the end devices become more distant from their current connected fog server, their application needs to be migrated to a more favourable one towards meeting its quality of service (QoS) requirements (e.g., latency) and improve their quality of experience (QoE). This transference can be achieved through migration capabilities of VMs, which can be used to move applications and data through fog nodes according to the device mobility (i.e. location). Summing up, in this context, virtualization is a vital technology at different levels namely: (1) isolation between untrusted user-level computations, (2) mechanisms for authentication, access control, and metering, (3) dynamic resource allocation for user-level computations, (4) the ability to support a very wide range of user-level computations, with minimal restrictions on their process structure, programming languages or operating systems [12], (5) mobility, migration and tasks offloading mechanisms, (6) power efficiency and (7) fault tolerance.

1.1 Motivation

Despite the benefits that fog promises to offer such as low latency, heterogeneity, scalability and mobility, the current model suffer from some limitations that still require efforts to overcome them.

There is lack of support for mobile fog computing. Most of the existing literature assumes that the fog nodes are fixed, or only considers the mobility of IoT devices [11]. Less attention has been paid to mobile fog computing and how it can improve the QoS, cost, and energy consumption. For instance, a bus could have computational power; as a fog node, it could provide offloading support to both end devices (inside and outside it) and other fog servers. The same could be applied to cars that are nowadays getting increasingly better in terms of computational power. Both would be extremely useful to enhance the resources and capabilities of fog computing. Specially in environments such as large urban areas, where traffic congestion is frequent or when those are parked (e.g., while an electric vehicle is charging). On top of that, it would reduce the implementation costs since it would no longer require such computational power in the fixed fog nodes. Finally, would reduce the costs to the client both in terms of latency and energy consumption since the fog nodes where they are connected may be even closer.

Another limitation of fog computing is to take into account few parameters in the decision making of migration. Most of the existing schemes that are proposed for fog systems, such as offloading, load balancing, or service provisioning, only consider few objectives (e.g., QoS, cost) and assume other objectives do not affect the problem [11]. Fog servers are less powerful than clouds due to the high deployment cost. If many requests are made to the same fog node at the same time, it will not have enough computational and storage power to give a prompt response. So it raises the question: textitshould a service currently running in one fog node be migrated to another one, and if yes, where? While conceptually simple, it is challenging to make these decisions in an optimal manner. Offloading tasks to the next server (i.e. upstream server) seems to be the solution, however, migrate the VM that was initially one-hop away from the IoT device to a multi-hop away server, will increase the network distance. Consequently, raises the end-to-end latency and the bandwidth usage by the intermediate links. Besides, this decision still has to take into account the cost for both the client (e.g., migration time, computational delay) and the provider (e.g., computing and migration energy). Ignoring some of these parameters can lead to wrong decisions that will both violate latency constraints of users' applications and damage or defeat the credibility of fog computing.

1.2 Objectives

Summing up, this work intend to tackle two of the current limitations that are little or no treated in the literature. One is to provide mobility support in fog computing environments, not exclusively to the end devices but also to the fog nodes and the other is to achieve multi-objective fog system design. Those objectives shall be implemented in a toolkit allowing the simulation of resource management techniques in IoT and mobile fog computing environments. In order to achieve the aforementioned goals, this work involves studying the current mobility approaches, publicly available, with respect to IoT or fog nodes. Also, analyze several optimization algorithms adopted in the field of data placement, propose our own architecture and target the toolkit in which we want to implement it.

The remainder of the document is structured as follows. Section 2 xxx. Section 4 describes xxxx. Section 5 defines the xxx. Finally, Section 6 presents xxxx and Section 7 xxxx.

2 Related Work

The solution proposed in this document leverages knowledge obtained from studying several concepts and systems from the current state-of-the-art. In this section, an overview of those concepts and systems will be given, stating for each of them their advantages and disadvantages. This section is structured as follows. Section 2.1 presents different methods to push intelligence and computing power closer to the source of the data and why this work adopted fog computing for this purpose. Section 2.5 presents mobility-aware systems xxx. Section **??** xxx. 2.4 xxx. 2.6 xxx. Section 2.7 xxx.

2.1 Related computing paradigms

In what concerns about standardizing fog computing, there is a lack of unanimity. As aforementioned, fog has been variously termed as cloudlets, edge computing, etc. Different research teams are proposing many independent definitions of fog (and fog-related computing paradigms). As there is a research gap in the definitions and standards for fog computing, this work follows the definitions that Ashkan Yousefpour et al. [11] present. Below, are described some paradigms that were raised in order to bring cloud closer to the end devices, as well as their pros and cons. As a conclusion we show why fog computing is the natural platform for IoT.

2.1.1 Mobile Computing

Mobile/nomadic computing (MC) is characterized by the processing being performed by mobile devices (e.g., laptops, tablets, or mobile phones). It raises to overcome the inherent limitations of environments where connectivity is sparse or intermittent and where there is low computing power. As this model only uses mobile devices to provide services to clients, there is no need for extra hardware. They already have communication modules such as Bluetooth, WiFi, ZigBee, etc. As already mentioned, mobile devices have evolved in recent years. However, their resources are more restricted, compared to fog and cloud. This computing paradigm has the advantage of being characterized by a distributed architecture, once mobile machines do not need a centralized location to operate. The disadvantages of MC are mainly due to their hardware nature (i.e. low resources, balancing between autonomy and the dependency of other mobile devices; characteristic that prevails in all distributed architectures) and the need of mobile clients to efficiently adapt to changing environments [13]. MC alone may not be able to meet the requirements of some applications. It is limited on the one hand due to autonomy constraints and in the order hand by low computational and storage capacity. This restricts the applications where this paradigm is feasible. For instance, it is unsuitable for applications that require low-latency and that, at the same time, generate large amounts of data that needs to be stored or processed. Nonetheless, MC can use both fog and cloud computing to enhance its capacities, not being restricted to a local network; expanding the scope of mobile computing and the number of applications where it can be used.

2.1.2 Mobile Cloud Computing

Cloud and fog computing, as mentioned in 2.1.1, are key elements for validate the importance of MC. This interaction between them results in a new paradigm, called mobile

cloud computing (MCC). MCC, differs from MC in the sense that mobile applications can be partitioned at runtime so that computationally intensive components of the application can be handled through adaptive offloading [14], from mobile devices to the cloud. This characteristic increases the autonomy of mobile devices (i.e battery lifetime), as both the data storage and data processing may occur outside them. Also, it enables a much broader range of mobile subscribers, rather than the previous laptops, tablets, or mobile phones. Opposed to resource-constrained in MC, MCC has high availability of computing resources, scaling the type of applications where it is possible to use (e.g., augmented reality applications). Unlike MC, MCC relies on cloud-based services, where its access is done through the network core by WAN connection, which means that applications running on these platforms require connection to the Internet all the time. On the one hand, both MCC and MC suffer from the intrinsic characteristics of mobility, such as frequent variations of network conditions (intensified under rapid mobility patterns), and on the other hand, in MCC, even if the mobile devices remain fixed, it suffers from the inherent disadvantage of using cloud-based services (i.e. communication latency), which makes it unsuitable for some delay-sensitive applications with heavy processing and high data rate.

2.1.3 Mobile ad hoc Cloud Computing

In some scenarios there exists lack of infrastructure or a centralized cloud, so implement a network based on MCC may not always be suitable. To overcome dependence on an infrastructure, raises mobile ad hoc cloud computing (MACC). It consists on a set of mobile nodes that form a dynamic and temporary network through routing and transport protocols. These nodes are composed by mobile ad hoc devices which may continuously join or leave the network. In order to counteract the aforementioned characteristics inherent to this type of networks, and unlike MC, a set of ad hoc devices may form a local cloud that can be used in the network for purposes of storage and computation. As mobile ad hoc networks (MANET), it is imperative in use cases such as disaster recovery, car-to-car communication, factory floor automation, unmanned vehicular systems, etc. Although it does not rely on external cloud-based services as MCC does, which mitigates the latency problem, it shares some limitations inherent to MC and ad hoc networks such as the power consumption constraints. Moreover, the formed local cloud may still be computationally weak and, as both network and cloud are dynamic it is more challenging to achieve an optimal performance (i.e. as there is no infrastructure, mobile devices are also responsible for routing traffic among themselves).

2.1.4 Edge Computing

Edge computing (EC), makes use of connected devices at the edge of the network to enhance its capabilities (i.e. management, storage, and processing power). It is located in the local IoT network, being ideally at one hop away from the Iot device and at most located few hops away. Open Edge Computing defines EC as computation paradigm that provides small data centers (edge nodes) in proximity to the users, enabling a dramatic improvement in customer experience through low latency interaction with compute and storage resources just one hop away from the user [15]. As in EC the connected devices don't have to wait for a centralized platform to provide the requested service, nor are so limited in terms of resources as in the traditional MC, their service availability is relatively high. Also, the restrictions over the autonomy are not so tight once there are not only mobile devices. Nonetheless, EC has some drawbacks. As latency, in this context, is composed by three components: data

sending time, processing time and result receiving time, even though the communication latency is negligible, processing time may not be. This computing paradigm only uses edge devices, and their computation and storage power may still be poor (e.g., routers, switches), compared to fog or cloud computing, so this processing latency may still be too high for some applications.

OpenFog Consortium states that fog computing is often erroneously called edge computing, but there are key differences between the two concepts [16]. Although they have similar concepts, edge computing tends to be limited to the edge devices (i.e. located in the IoT node network), excluding the cloud from its architecture. Whereas, fog computing is hierarchical and unlike EC, it is not limited to a local network, but instead it provides services anywhere from cloud to things. It is worth noting that the term edge used by the telecommunication's industry usually refers to 4G/5G base stations, radio access networks (RANs), and internet service provider (ISP) access/edge networks. Yet, the term edge that is recently used in the IoT landscape refers to the local network where sensors and IoT devices are located [11].

2.1.5 Multi-access Edge Computing

Analogously, MCC is an extension of MC through CC, as multi-access edge computing (MEC) is an extension of MC through EC (telecommunication industry definition). ETSI defines MEC as computation paradigm that offers application developers and content providers cloud-computing capabilities and an IT service environment at the edge of the network. This environment is characterized by ultra-low latency and high bandwidth as well as real-time access to radio network, information that can be leveraged by applications [17]. In MEC, operators can open their RAN edge to authorized third parties, allowing them to deploy applications and services towards mobile subscribers through 4G/5G base stations. The first approach to the edge of a network meant the edge of a mobile network, hence the name mobile edge computing. As MEC research progressed, was noticed that the term leaves out several access points that may also construct the edge of a network. Thus, prompted the change from mobile edge computing to multi-access computing in order to reflect that the edge is not solely based on mobile networks [18]. Now it includes a broader range of applications beyond mobile device-specific tasks, such as video analytics, connected vehicles, health monitoring, augmented reality, etc. Similar to EC, MEC can operate with little to no Internet connectivity and use small-scale data centers with virtualization capacity to provide services. MEC is expected to benefit significantly from the up-and-coming 5G platform as it allows for lower latency and higher bandwidth among mobile devices, and it supports a wide range of mobile devices with finer granularity.

Both fog computing and MEC have the objective of offering similar type of features. Fog computing concentrates on applications, mainly IoT, that take advantage of a platform set that collectively assist end devices. MEC, on the other hand, focuses on application-related enhancements in terms of feedback mechanisms, information and content processing and storage, etc [19].

2.1.6 Cloudlet Computing

Cloudlet computing is another direction in mobile computing that aims to bring cloud closer to end devices through the use of cloudlets. M. Satyanarayanan et al. states that a cloudlet is a trusted, resource-rich computer or cluster of computers that's well-connected

to the Internet and available for use by nearby mobile devices [20]. Cloudlet is, as the name suggests, a smaller sized clouds with lower computational capacity. It can be seen as a "data center in a box", where mobile users can exploit their virtual machines (VM) to rapidly instantiate customized-service software in a thin client fashion. This way, it is possible to offload computation from mobile devices to VM-based cloudlets located on the network edge (telecommunication industry definition). Through those VMs, cloudlets are capable of providing resources to end devices in real-time over a WLAN network. The relatively low hardware footprint, results in moderate computing resources, but lower latency and energy consumption and higher bandwidth compared to cloud computing. The characteristics of this computing paradigm make possible to handle applications with low-latency requirements, supporting real-time IoT applications. Y Jararweh et al. [21] propose an architecture mobile-cloudlet-cloud, where they present three reasons which indicate that even though cloudlets are computationally powerful, they still need a connection to the cloud and its services: (1) Heavy non real time jobs might process in the enterprise cloud whiles the real-time ones processed by the cloudlet, (2) Accessing a file stored in the Enterprise Cloud, (3) Accessing some services that are not available inside the Cloudlet. Although cloudlet computing fits well with the mobile-cloudlet-cloud architecture, fog computing offers a more generic alternative that natively supports large amounts of traffic, and allows resources to be anywhere along the cloud-to-things continuum. As it will be shown later, cloudlets are great resources and, in this way, they can be combined with the fog computing paradigm.

2.1.7 Mist Computing

Mist computing emerges to push IoT analytics to the extreme edge. This computing paradigm is an even more dispersed version of fog. That means locating analytics tools not just in the core and edge, but at the "extreme edge" [22]. Mist computing layer is composed by mist nodes that are perceived as lightweight fog nodes. They are more specialized and dedicated nodes with low computational resources (e.g., microcomputers, microcontrollers) that are even closer to the end devices than the fog nodes [23]. Therefore, mist computing can be seen as the first (non-mandatory) layer in the IoT-fog-cloud continuum. It extends compute, storage, and networking across the fog through the *things*. This decreases latency and increases subsystems' autonomy. It can be implemented in order to enhance the services of predominance of wireless access and mobility support. The challenge with implementing mist computing systems lies in the complexity and interactions of the resulting network. These must be managed by the devices themselves as central management of such systems is not feasible.

2.1.8 Concluding Remarks

Just like the above-mentioned, there are some other similar computing paradigms such as follow me cloud (FMC), follow me edge (FME), follow me edge-cloud (FMeC) and cloud of things (CoT), to name a few. However, this state-of-the-art had as first objective to investigate the most relevant and treated ones in the literature. The purpose was to understand their characteristics and identify where to tackle the current limitations which oppose the deployment of delay-sensitive IoT systems in mobile environments. Fig. 1 shows a classification of the these paradigms and their overlap in terms of their scope.

As shown with the attributed pros and cons to these computing paradigms, they all have been proposed to cover different use cases. Even so, fog computing is suited for many

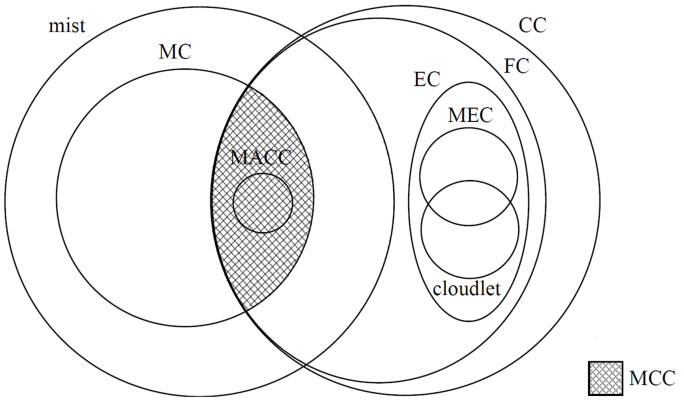

Fig. 1: A classification of scope of fog computing and its related computing paradigms.[1]

use cases, including data-driven computing and low-latency applications, being the most versatile and comprehensive one. As aforementioned, fog is flexible enough to interact and take advantage of other paradigms such as edge, cloud, cloudlet and mist computing. Nonetheless, it may not be suitable for a few extreme use cases, such as disaster zones or sparse network topologies where ad hoc computing (e.g., MACC) may be a better fit.

2.2 Fog computing architecture

Fog computing is a great resource to support IoT applications' requirements. Taking into account what has been mentioned in Section 1 and Section 2.1, fog computing has, the below, eight fundamental characteristics which validate the statement uttered above [23]:

- **Contextual location awareness, and low latency**. Provides low latency due to the proximity between the IoT devices and the fog nodes. Also, the contextual location allows them to be aware of the cost of communication latency with both other fog nodes and the end devices, allowing the distribution of applications across the network to be organized in a weighted manner;
- **Geographical distribution**. Uses anything between the cloud and *things* to provide ubiquitous computing, allowing continuity of service in mobile environments;
- **Heterogeneity**. Supports wide diversity of communications, applications and services;
- **Interoperability and federation**. Uses cooperation of different providers to support heavy applications such as real-time streaming. Moreover, it supports migration of applications to more suited fog servers depending on the current context;
- **Real-time interactions**. Applications may involve real-time interactions rather than batch processing (e.g., as cloud does);
- **Scalability and agility of federated, fog-node clusters**. Fog is adaptive; may form clusters-of-nodes or cluster-of-clusters to support elastic compute, resource pooling, etc;
- **Predominance of wireless access**. Most of the end devices only support wireless communication;
- **Support for mobility**. The exponential growth of mobile devices demands mobility support.

[1] Figure adapted from [11].

Nonetheless, fog still has some limitations (as stated in Section 1.2). In order to tackle those limitations, first it is needed an overview over its architecture. This includes understanding what are the actors and how they interact, how IoT nodes connect to the fog servers, how clients outsource the allocation and management of resources that they rely upon to these servers, how the location-aware and migration features are performed, etc.

2.2.1 Overview

Fog computing is composed by fog nodes/servers, that allow the deployment of distributed, latency-aware applications and services. Those nodes can be either virtual (e.g., gateways, switches, routers, servers) or physical (e.g., virtualized switches, virtual machines, cloudlets) components that provide computing resources to end devices. They can be organized in clusters either vertically (to support isolation), horizontally (to support federation), or relative to fog nodes' latency-distance to the IoT devices [23]. Fog nodes can be accessed through connected devices located at the edge, which provide local computing resources and, when needed, provide network connectivity to centralized services (i.e. cloud). Moreover, fog nodes can operate in a centralized or decentralized manner and can be configured as stand-alone nodes.

Fig. 2 shows the typical fog computing architecture. As stated before, mist computing can be implemented in a layer between the fog servers and the end devices. Moreover, the presence of cloud servers is not imperative, however it is very important for numerous applications. It is worth noting that, once fog nodes can be anything with computational and storage power in the cloud-to-things continuum, the links formed in these architectures (i.e. End device-to-Fog, Fog-to-Fog and Fog-to-Cloud) can be of any type. For instance, end devices can be connected to fog servers by wireless access technologies (e.g., WLAN, WiFi, 3G, 4G, ZigBee, Bluetooth) or wired connection. Moreover, fog nodes can be interconnected by wired or wireless communication technologies and they can be linked into the cloud by IP core network. Nonetheless, in order to provide location awareness, the fog layer also needs Location-Based Services (LBSs) that provide location information.

In this architecture, the connected sensors located at the edge, generate data that can adopt two models. First, in a sense-process-actuate model, the information collected is transmitted as data streams, which is acted upon by applications running on fog devices and the resultant commands are sent to actuators. In this model, the raw data collected often does not need to be transferred to the cloud; data can be processed, filtered, or aggregated in fog nodes, producing reduced data sets. The result can then be stored inside fog nodes or actuated upon through the actuators. Second, in a stream-processing model, sensors send equally data streams, where the information mined (from the incoming streams) is stored in data centers for large-scale and long-term analytics. In this case, big data needs to be stored and does not have that much latency constraints. Being fog servers less powerful than the cloud ones, cloud is far more suited for this kind of operations. Yet, fog servers can still shrink data, doing some intermediate processing as in the previous model. This meets the aforementioned statement - although cloud is not essential for the functioning of fog, in some applications it is beneficial or even essential.

Fog servers are the fundamental components in this three tier architecture (i.e. IoT-fog-cloud). They are able to support the six features shown below [23]:

- **Autonomy**. Fog nodes can be autonomous enough to operate independently, making local decisions, at the node or cluster-of-nodes level;
- **Heterogeneity**. Can be deployed in a wide variety of environments;

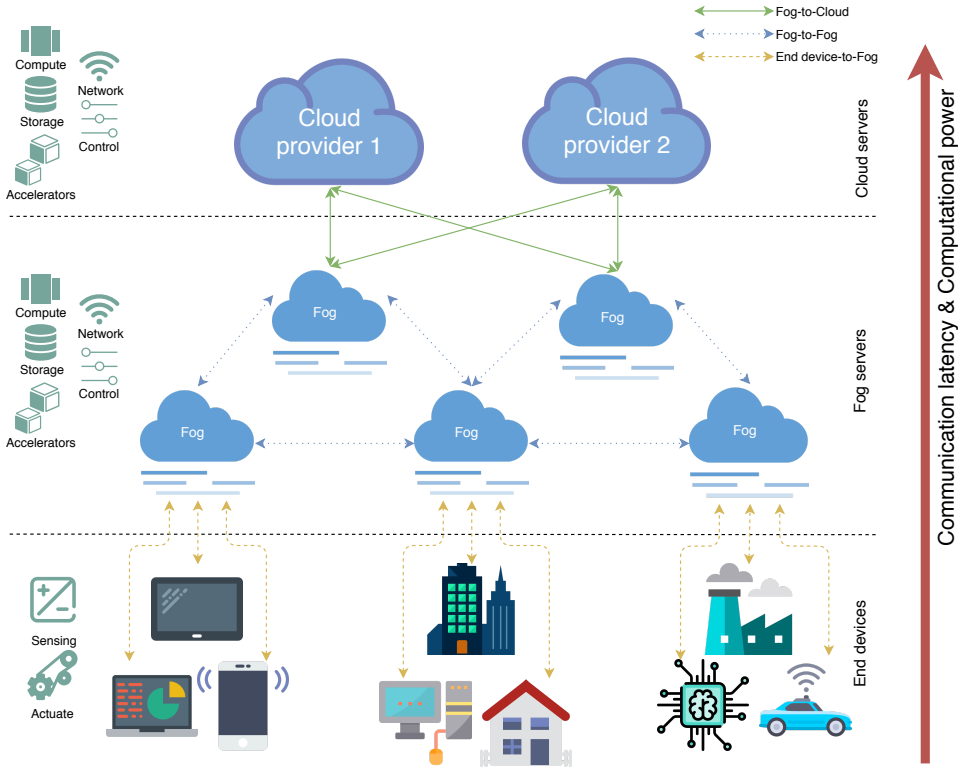

Fig. 2: Typical architecture of fog computing.

– **Hierarchical clustering**. The fog network can be organized with different numbers of layers, so that they are able to provide different subsets of service functions while working together as a continuum;
– **Manageability**. They are managed and orchestrated by complex systems that can perform most routine operations automatically;
– **Programmability**. Fog nodes are inherently programmable at multiple levels, by multiple stakeholders such as network operators, domain experts, equipment providers, or end users.

Fog nodes generally are of most value in scenarios where data needs to be collected at the edge and where the data from thousands or even millions of devices is analyzed and acted upon in micro and milliseconds [24]. In order to being able to support such a large number of requests, especially those engaged in enhanced analytics, fog nodes may implement additional hardware. Accelerators modules (refer to Fig. 2) can be implemented to provide supplementary computational throughput. For instance, hardware accelerators can be performed through Graphics Processing Unit (GPUs); they are an optimal choice for applications that support parallelism or for stream processing. Also, fog nodes can opt to make use of Field Programmable Gate Arrays (FPGAs) or even Digital Signal Processors (DSPs) for this propose.

2.2.2 Data Placement

Moreover, the applications deployed by the end users in fog nodes can be seen either as a whole or as a distributed data flow (DDF) model, in which the applications are moduled as a collection of modules. This can be particular useful so that the less restricted modules in terms of latency can be deployed to the upper fog layers (ideally to the cloud), leaving the fog nodes of the lower layers less overloaded, being able to respond faster to modules within tighter latency bounds. Nonetheless, fog nodes can communicate between them to perform data and process management in order to support application requirements, as well as to exchange fog control/management data such as user device and application state.

2.2.3 Visualization Mechanism

In general, hosting an application involves creating a set of virtual machines (VMs) or execution containers (e.g., Docker) and assigning them a vector of computing resources (such as CPU, memory and storage) from the physical machines (PMs) in the edge-cloud.

2.2.4 Orchestration

When an end device needs to offload some work to a third party, it needs somehow to know where to outsource the allocation and management of resources that they rely upon. For this propose, the fog architecture also needs a discovery and orchestration service which concerns in finding the best available fog server, given certain capabilities and requirements.

In this context, E Saurez et al. [27] propose foglets, a programming model that facilitates distributed programming across fog nodes. Foglets are implemented in fog nodes using container technology. They provide APIs for app development as a dataflow graph whose nodes can be placed in the different levels of the computational hierarchy. In additional to the provided primitives for communication between the application components, it also embodies algorithms for the discovery and incremental deployment of resources commensurate with the application needs. Moreover it provides mechanisms for QoS-sensitive and workload sensitive migration of application components due to end devices mobility and application dynamism. Specifically they show that there are four entities in the foglets runtime system: the discovery server, the docker registry server, the entry point daemon, and finally the worker process. The discovery server is a partitioned name server that maintains a list of fog nodes available. Docker registry server is a server that contains the binaries for the applications that have been launched on the foglets infrastructure. The entry point daemon executes directly on top of the host OS in the fog node, awaits requests and periodically sends "I am alive" message to the discovery server. Finally the worker process that will carry out the functionality contained in a particular application component assigned to it.

Also in this context, F Bonomi et al. [25] propose a fog computing layer architecture that is classified into two sub-layers: (1) the fog abstraction layer, and (2) the fog orchestration layer. While the former enables virtualization, provides data and resource isolation guarantees, manages the fog resources, and preserves security and privacy, the latter provides dynamic, policy-based life-cycle management of fog services. The fog orchestration layer comprises of a software agent, Foglet, with reasonably small footprint yet capable of

bearing the orchestration functionality and performance requirements. Moreover, this layer also needs a distributed, persistent storage to store policies and resource meta-data (e.g., capability, performance) that support high transaction rate update and retrieval, a scalable messaging bus to carry control messages for service orchestration and resource management and finally a distributed policy engine with a single global view and local enforcement.

Julien Gedeon [26], propose a brokering mechanism in which available surrogates (e.g., cloudlets) advertise themselves to the broker. The broker receives client requests and considers a number of attributes such as network information, hardware capabilities, and distance to find the best available surrogate for the client. They look at the problem of surrogate discovery in the context of an urban area, where they are faced with a high mobility of end devices. Multiple brokers are interconnected using Distributed Hash Tables (DHTs) in order to exchange information.

Fog servers can provide reduced latencies and help in avoiding/reducing traffic congestion in the network core. However, this comes at a price: more complex and sophisticated resource management mechanisms are needed. This raises new challenges to be overcome such as dynamically deciding when, and where (device/fog/cloud) to carry out processing of requests to meet their QoS requirements. Furthermore, in mobile environments such mechanisms must incorporate mobility (i.e. location) of data sources, sinks and fog servers in the resource management and allocation process policies to promote and take advantage of proximity between fog and users.

2.3 Data placement

[28] They propose a layered Fog framework to better support IoT applications through virtualization. The virtualization is divided into object virtualization (VOs), network function virtualization and service virtualization. VOs to address the protocol inconsistency (lack of unified networking protocols that leads to exaggerated overhead); Network function virtualization maps standard networking services to VOs, thus, minimize the communication process between consumers and producers by minimizing latency, improving security and scalability; Service virtualization that composes the community and Cloud Apps from various vendors to serve local Fog users with high quality of experience (QoE) but at low cost. At last, Foglets are involved to seamless aggregate multiple independent virtual instances, Fog network infrastructures, and software platforms.

[29] This paper proposes a Distributed Dataflow (DDF) programming model for the IoT that utilizes computing infrastructures across the Fog and the Cloud. Also, evaluate their proposal by implementing a DDF framework based on Node-RED (Distributed Node-RED or D-NR), a visual programming tool that uses a flow-based model for building IoT applications. To address challenges of the intrinsic nature of the IoT (heterogeneous devices/resources, a tightly coupled perception-action cycle and widely distributed devices and processing), they propose a Distributed Dataflow (DDF) programming model for the IoT that utilities computing infrastructures across the Fog and the Cloud. Also, they evaluate their proposal by implementing a DDF framework based on Node-RED (Distributed Node-RED or D-NR), a visual programming tool that uses a flow-based model for building IoT applications.

[30] The authors address the problem of multi-component application placement on fog nodes. Each application could be modeled as a graph, where each node is a component of the application, and the edges indicate the communication between them.

[31] With the state-of-the-art virtualization technologies, services can be implemented in modular software as a graph/chain of portable VOs that can be dynamically migrated around the Telco infrastructure. It is proposed a VO clustering and migration policy that jointly considers user proximity and inter-VO affinity to scalably support user mobility, while allowing service differentiation among users.

2.4 Migration optimization in mobile fog environments

When an IoT device needs to offload some heavy application to a third party, it will be connected to the nearest server, securing an one-hop away fog server to ensure the shortest network delay. However, as their physical distance increases either by device or server movement, their network distance (i.e. the number of hops) will also increase. Hence, both latency and bandwidth usage by the intermediate links will increase, resulting in poor connectivity. This away, in such dynamic environments the decision-making of where to send the VMs, in order to overcome these limitations, is a major concern. Moreover, even if both clients and servers are static, the end-to-end latency may increase due to unexpected crowds of mobile clients seeking to connect or making requests to the same fog server simultaneously.

In Section **??** was verified that applications can be offloaded as a whole, or as a set of modules that may have different constraints. Regardless of their type, whenever it is justified the system needs to be readjusted. This is performed through the exchange of VMs (that contain the applications or modules) between fog nodes. For this reason, it is necessary to answer the following two questions: *When is this exchange justified? And what is the best placement for those applications and/or modules?*

2.4.1 Latency-aware with mobile end-devices

In the context of mobile end devices, B Ottenwälder et al. [32] state that as sensors and consumers are mobile, the latency (between the access point of a mobile sensor and the fog node) and bandwidth usage is expected to change over time, so it is necessary to constantly adapt the placement through migrations to new fog nodes. However, each migration comes with a cost; consequence of the local state that also needs to be migrated. Thus, frequent migration would significantly decrease the system performance. To overcome this limitation, they propose a placement and migration method for providers of infrastructures that incorporate cloud and fog resources to support operator migrations in mobile complex event processing (MCEP) systems. Their method plans the migration ahead of time through knowledge of the MCEP system and predicted mobility patterns towards ensuring application-defined end-to-end latency restrictions and reducing the network utilization. These predicted mobility patterns were captured using three different methods: uncertain locations from the *dead reckoning* approach (linear), certain locations that could stem from a *navigation* system (navi), and *learned* transitions between leaf broker (learned). This method, allows a minimization of migration costs by selecting migration targets that ensure a low expected network utilization for a sufficiently long time. Moreover, they present how the application knowledge of the CEP system can be used to improve current live migration techniques for VMs to reduce the required bandwidth during the migration (i.e. unnecessary events are not migrated).

A different approach was adopted in the work of R Urgaonkar et al. [33]. Similarly to the previous work, their aim is to provide an optimal decision with regard to: where to migrate a current service as the user location changes. However, they argue that because of the uncertainty in user mobility and request patterns, it is challenging to make the decision in an optimal manner. Also, in this work is argued that methods that depend on mobility patterns have several drawback, namely: (1) it requires extensive knowledge of the statistics of the user mobility and request arrival processes that can be impractical to obtain in a dynamic network, (2) even when this is known, the resulting problem can be computationally

challenging to solve, and (3) any change in the statistics would make the previous solution suboptimal and would require recomputing the optimal solution. Thus, they propose a new model, inspired by the technique of Lyapunov optimization, that overcomes these drawbacks (i.e. does not require any knowledge of the transition probabilities). The overall problem of dynamic service migration and workload scheduling to optimize system cost while providing end user performance guarantees is formulated as a sequential decision making problem in the framework of markov decision problems (MDPs). However, they have developed a new approach for solving a class of constrained MDPs that possess a decoupling property. When this property holds, their approach enables the design of simple online control algorithms that do not require any knowledge of the underlying statistics of the MDPs.

W Zhang et al. [34] state that previous studies have proposed a static distance-based MDP for optimizing migration decisions. However, these models fail to consider dynamic network and server states in migration decisions, assuming that all the important variables are known. Moreover, they also point out another unaddressed problem which lies in the recalculation time interval of the method. Since running MPD is a heavy computing task, a short recalculation interval introduces a considerable overhead to the server. On the other hand, a long recalculation interval may translate into lazy migration, meaning that resulting in periods of transgression of QoS guarantees. In order to overcome those issues, the authors propose SEGUE. This model achieves optimal migration decisions by providing a long-term optimal QoS to mobile users in the presence of link quality and server load variation. Additionally, SEGUE adopts a QoS aware scheme to activate the MDP model. In other words, it only activates the MDP model when QoS violation is predicted. Thus, it avoids unnecessary migration costs and bypass any possible QoS violations while keeping a reasonable low overhead in the servers.

The work performed by Wuyang Zhang et al. [35] use as case study the Massively Multiplayer Online Gamse (MMOGs) with Virtual Reality (VR) technologies, VR-MMOGs. They present the main challages of VR-MMOGs, namely: stringent latency, high bandwidth, and large scale requirements. This work shows one problem that remains unsolved: how to distribute the work among the user device, the edge clouds, and the center cloud to meet all three requirements especially when users are mobile. Their approach was to place local view change updates on edge clouds for immediate responses, frame rendering on edge clouds for high bandwidth, and global game state updates on the center cloud for user scalability. In this kind of games, the users need to move, so in order to keep a low latency communication, they also ropose an efficient service placement algorithm based on MDP. This method takes into account the presence of dynamic network states and server workload states, and user mobility. To ensure feasibility of this method, they come up with an approach that reduces the algorithm complexity in both storage and execution time. Nonetheless, unlike many of the service migration solutions which assumes an ignorable service transition time, they point out that it is impossible to migrate an edge service from one edge to another instantly given the size of a VR game world. Therefore, they propose a mechanism to ensure a new edge cloud is activated when a player connects to the new one.

S Abdelwahab et al. [36] argue that IoT devices communicate a large number of messages with many devices. Thus, devices with low computing and storage capacities will became another source of latency for large-scale distributed applications. Their experiments show that brokering the messages through a one-hop away broker may reduce significantly

the end-to-end latency. Therefore, if devices are cloned in a one-hop away cloudlet, their clones can provide message brokering service, allowing both a communication with minimal latency between devices and to offload intensive computation into rich memory and processing nodes that host the clones. Nonetheless, communicating through a one-hop away clone may still experience long end-to-end latency when the broker service relays messages to distant devices. Hence, they propose FogMQ, a self-deploying brokering clones that discover hosting platforms and autonomously migrate between them according to the measured end-to-end latency. This method does not need a central monitoring and control unit. FogMQ servers expose tomography functionalities that enables clones to take migration decisions without complete knowledge about the hosting platform. It allows to stabilize clones deployment and achieve a near minimum latency given an existing infrastructure limits.

The study performed by X Sun et al. [37] presents, similarly to the previous ones, a case scenario where end devices are mobile. To preform this work they use a cloudlet network architecture to bring the computing resources from the centralized cloud to the edge. They present the PRofIt Maximization Avatar pLacement (PRIMAL) strategy. PRIMAL maximizes the trade-off between the migration gain (i.e. the end-to-end delay reduction) and the migration cost (i.e. the migration overheads), selectively migrating the avatars (an application clone located in a cloudlet) to their optimal locations.

2.4.2 Latency-aware with mobile fog servers

D Ye et al. [38] leverage the characteristics of buses and propose a scalable fog computing paradigm with servicing offloading in bus networks. Knowing that buses have fixed mobility trajectories and strong periodicity, they consider a fog computing paradigm with service offloading in bus networks which is composed by two parts: roadside cloudlets and bus fog servers. The roadside cloudlet consists of three components: dedicated local servers, location-based service (LBS) providers, access points (APs). The dedicated local servers virtualize physical resource and act as a potential cloud computing site. The LBS providers offer the real time location of each bus in bus networks. The APs act as gateways for mobile users and bus fog servers within the communication coverage to access the roadside cloudlet. When users need to offload some computationally intensive and delay sensitive tasks, they access APs and use the computing service of the roadside cloudlet. However, as cloudlets have limited computational and storage resources, they may became overloaded. The bus fog server is a virtualized computing system on bus, which is similar to a light-weight cloudlet server. Hence, those buses not only provide fog computing services for the mobile users on bus, but also are motivated to accomplish the computation tasks offloaded by roadside cloudlets. This allocation strategy is accomplished using genetic algorithm (GA), where the objective is to minimize the cost that roadside cloudlets spend to offload their computation tasks. Meanwhile, the user experience of mobile users are maintained. Although this work refers to mobile users, its meaning is not literal, being supported only the mobility of fog servers. In their problem formulation there are two types of mobile users. On one hand there are mobile users that already have offloaded their computing tasks to the roadside cloudlets (i.e. representing the workload of the cloudlets). On the other hand there are several mobile users inside the bus that have also offloadded their tasks (i.e. representing the workload of bus fog servers).

2.5 Mobile Fog Computing

The concept of mobile fog computing is similar to fog computing, in which both IoT and fog nodes are mobile components. Those are connected wirelessly (e.g., via WiFi or Bluetooth). The challenge with implementing fog computing in mobile environments lies in the underlying complexity of data placement management and decision-making to ensure the QoS to all users (i.e. ensure that all latency constraints of users' applications are met).

In this context, Luiz F. Bittencourt et al. [39] took into account the same architecture shown in Fig. 2. With the use of two applications (electroencephalography (EEG) tractor beam game to test near-real-time applications and video surveillance / object tracking application for delay-tolerant applications) to test tree different scheduling strategies (Concurrent, the First Come-First Served (FCFS), and the Delay-priority strategies), and check how scheduling decision and the change of cloudlet by the players impact the network traffic and the delays.

MM Lopes et al. [40] discuss resource allocation in fog computing in the face of users' mobility, where mobility is achieved through migration of virtual machines between cloudlets. They present a new migration technique composed of two modules: migration policy which defines when the user VM should be migrated, considering aspects such as the user's speed, direction and geographical position and migration strategy, the destination cloudlet, and how the migration is performed. This work had the objective of study the impact of different migration strategies in the latency with users' mobility.

2018 **Dynamic Mobile Cloudlet Clustering for Fog Computing.** Fog Computing is one of the solutions for offloading the task of a mobile. However the capability of fog server is still limited due to the high deployment cost. In this paper, is proposed a dynamic mobile cloudlet cluster policy (DMCCP) to use cloudlets as a supplement for the fog server for offloading. The main idea is that by monitoring each mobile device resource amount, the DMCCP system clusters the optimal cloudlet to meet the requests of different tasks from the local mobile device.

Although several studies were already done in order to provide mobile support for IoT devices, the purpose of this study is to support mobile fog computing, once fog nodes can be anything in the path that connects things to the cloud. This distributed middle tier, in the 3-tier architecture (things-fog-cloud), can use as fog nodes any physical device that has facilities or infrastructures that can provide resources and visualization capabilities. This, may include movable fog nodes, such as cars, buses, unmanned aerial vehicles (UAVs), etc. The importance of mobile fog nodes cannot be overlooked, once they may represent a way to offload fixed cloudlet tasks and thus improve fog features. In this field there are already some early efforts.

Dongdong Ye et al. [38] show a use case where buses are used as mobile cloudlets. They leverage the characteristics of buses (e.g., the same routes, many stops) and propose a scalable fog computing paradigm with servicing offloading in bus networks. The bus fog servers not only provide fog computing services for the mobile users on bus, but also are motivated to accomplish the computation tasks offloaded by roadside cloudlets. By this way, the computing capability of roadside cloudlets is significantly extended.

[172] 2016 **Vehicular fog computing: A viewpoint of vehicles as the infrastructures.** Xueshi Hou et al. present the idea of utilizing vehicles as the infrastructures for communication and computation, named vehicular fog computing (VFC), which is an architecture that utilizes a collaborative multitude of end-user clients or near-user edge devices to carry out communication and computation, based on better utilization of individual communication and computational resources of each vehicle. They discussed four types of scenarios of moving and parked vehicles or congested traffic. Also, they point out the advantages against vehicular cloud computing (VCC) and the advantages in scenarios like of emergency operations for natural disaster and terrorist attack.

[186] 2018 **Mobile edge computing via a uav-mounted cloudlet: Optimization of bit allocation and path planning.** Unmanned aerial vehicles (UAVs) have been considered as means to provide computing capabilities. In this model, UAVs act as fog nodes and provide computing capabilities with enhanced coverage for IoT nodes. The system aims at minimizing the total mobile energy consumption while satisfying QoS requirements of the offloaded mobile application. This architecture is based on a UAV-mounted cloudlet which provides the offloading opportunities to multiple static mobile devices. They aim to minimize the mobile energy consumption, while satisfying QoS requirements and optimize UAV's trajectory.

[270] 2016 **An adaptive cloudlet placement method for mobile applications over gps big data.** Introduces the concept of movable cloudlets and explores the problem of how to cost-effectively deploy these movable cloudlets to enhance cloud services for dynamic context-aware mobile applications. To this end, Haolong Xiang et al. propose an adaptive cloudlet placement (via GPS) method for mobile applications. Specifically, the gathering regions of the mobile devices are identified based on position clustering, and the cloudlet destination locations are confirmed accordingly. Besides, the traces between the origin and destination locations of these mobile cloudlets are also achieved.

2.5.1 Handover

X Sun et al. [41] shows an architecture where each User Equipment (UE) has its own Avatar (a private computing and storage resources for the UE) which is deployed to a cloudlet, being the communication characterized by low end-to-end (E2E) latency. When UEs roam away, in order to maintain the end-to-end latency, their Avatars should be handed off among cloudlets accordingly. However, moving such amount of data (the Avatar's virtual disk) during the handoff time may both incur unbearable migration time and network congestion. In order to overcome those limitations, they propose LatEncy Aware Replica placemeNt (LEARN) algorithm to place a number of replicas of each Avatar's virtual disk into suitable cloudlets. Meanwhile, by considering the capacity limitation of each cloudlet, they propose the LatEncy aware Avatar hanDoff (LEAD) algorithm to place UEs' Avatars among the cloudlets such that the average E2E delay is minimized.

[42] The authors observe that traditional mobile network handover mechanisms cannot handle the demands of fog computation resources and the low-latency requirements of mobile IoT applications. The authors propose Follow Me Fog framework to guarantee service continuity and reduce latency during handovers. The key idea proposed is to continuously

monitor the received signal strength of the fog nodes at the mobile IoT device, and to trigger pre-migration of computation jobs before disconnecting the IoT device from the existing fog node.

[43] Present a novel service handoff system which seamlessly migrates offloading services to the nearest edge server, while the mobile client is moving. Service handoff is achieved via container migration. They have identified an important performance problem during Docker container migration, proposing a migration method which leverages the layered storage system to reduce file system synchronization overhead, without dependence on the distributed file system.

[44] Develop a novel user-centric energy-aware mobility management (EMM) scheme, in order to optimize the delay, under energy consumption constraint of the user. Based on Lyapunov optimization and multi-armed bandit theories, EMM works in an online fashion. Theoretical analysis explicitly takes radio handover and computation migration cost into consideration and proves a bounded deviation on both the delay performance and energy consumption compared with the oracle solution with exact and complete future system information. The proposed algorithm also effectively handles the scenario in which candidate BSs randomly switch ON/OFF during the offloading process of a task.

[45] Study of mobility support issue in fog computing for guaranteeing service continuity. Propose a novel SDN enabled architecture that is able to facilitate mobility management in fog computing by decoupling mobility management and data forwarding functions. Design an efficient handover scheme by migrating mobility management and route optimization logic to the SDN controller. By employing link layer information, the SDN controller can pre-compute the optimal path by estimating the performance gain of each path.

[46] To guarantee the strict latency requirements, new solutions are required to cope with the user mobility in a distributed edge cloud environment. The use of proactive replication mechanism seems promising to avoid QoE degradation during service migration between different edge nodes. However, accounting for the limited resources of edge micro data-centers, appropriate optimization solutions must be developed to reduce the cost of service deployment, while guaranteeing the desired QoE. In this paper, Ivan Farris et al., by leveraging on prediction schemes of user mobility patterns, have proposed two linear optimization solutions for replication-based service migration in cellular 5G networks: the min-RM approach aims at minimizing the QoE degradation during user handover; min-NSR approach favors the reduction of service replication cost. Simulation results proved the efficiency of each solution in achieving its design goal and provides useful information for network and service orchestrators in next-generation 5G cloud-based networks.

2.6 Multi-objective

In section 2.4 the focus was in minimizing end to end latecy and the bandwith usage however those objectives are not everything. we need to keep in mind that migration also brings costs for the providers... (ver o resto dos problemas abaixo e os que são apresentados na introdução) QoS, QoE, Cost, Energy, Handover, Mobility, Bandwidth

2.7 Toolkits

As stated before, in Section 1.2, the proposed solution, which will be described later in Section 4, will be implemented in a carefully selected toolkit. In order to make this selection, a survey was made on the currently available simulators. Table 1 compares fog and related computing paradigms simulators via comparison of their characteristics.

Programming language: This is important to evaluate the simplicity, level of abstraction offered, maintainability, extensibility, its popularity, etc. As can be observed, almost all are Java-based, being that all opt for object-oriented programming.

Availability: All presented simulators are publicly available, except for RECAP (a H2020 project to develop the next generation of cloud, edge and fog computing capacity provisioning via targeted research advances in cloud infrastructure optimization, simulation and automation) that is still in progress.

Documentation: Unlike the availability, documentation is not always available or sometimes is scarce. In those cases, this is an impediment to the extensibility and maintenance of the corresponding simulators. This parameter includes official documentation, tutorials, community, wiki, etc

Graphical support: Provide a Graphical User Interface (GUI) may be helpful. Instead of defining the entire architecture programmatically, researchers can define it in a user-friendly environment.

Energy-aware: As aforementioned, energy is one of the multi-objectives that this work intends to provide. When implementing the migration optimization algorithm, the more realistic the energy model, the more realistic the algorithm will be. Although CloudSim provides energy-conscious resource management techniques/policies, GreenCloud is a more fine-grained simulator to this end. Its energy models are implemented for every data center element (computing servers, core and rack switches). Moreover, due to the advantage in the simulation resolution, energy models can operate at the packet level as well. This allows updating the levels of energy consumption whenever a new packet leaves or arrives from the link, or whenever a new task execution is started or completed at the server [47].

Cost-aware: Similarly, cost-aware is also an important parameter in the migration optimization algorithm. Although one of the main objectives of fog computing is to guarantee QoS to its users in mobile environments, migration of applications or its modules have associated costs. Those are related with the increase of bandwidth usage and the energy spent in the transmission. Thus, a trade-off between QoS and cost has to be obtained. Moreover, migration results in an increase usage of computing resources that are performing non-useful work (overhead). Therefore, a cost model refers to the quantification in monetary terms of the use of the above mentioned parameters. This is important because pay-as-you model go is one of fundamental services of both cloud and fog computing.

Application models: This is an important feature in terms of QoS because it allows specifying the computational requirements for the application and a specific completion deadline.

Communication model: CloudSim can model network components, such as switches, but lacks fine-grained communication models of links and Network Interface Cards (NIC) causing VM migration and packet simulation to be network-unaware [48]. CloudNetSim++, on the other hand, supports a simulation model of real physical network characteristics such as network congestion, packet drops, bit error, and packet error rates. Moreover, GreenCloud allows communications based on TCP/IP protocol. It allows capturing the dynamics of widely used communication protocols such as IP, TCP, UDP, etc. Whenever a message needs to be transmitted between two simulated elements, it is fragmented into a num-

ber of packets bounded in size by network Maximum Transmission Unit (MTU). Then, while routed in the data center network, these packets become a subject to link errors or congestion-related losses in network switches [47].

Migration support: This policy allows to apply data placement techiniques (i.e. application and workload migration) to benefit high quality of service.

Mobility/Location-aware: As already explained, mobility/location-aware is quite an essential feature in fog computing. This allows maintaining (as much as possible) the E2E latency as both users and servers move. There are few simulators that support this feature. For instance, in a cloud environment, CloudAnalyst [49] is a tool whose goal is to support the evaluation of social network applications, according to the geographic distribution of users and data centers. On the contrary, in a fog environment, to the best of our knowledge, there are no support for geographic distribution of fog nodes. The only one that provides mobility/location-aware of end devices, which is currently available, is MyiFogSim. MyiFogSim is an extension of iFogSim to support users mobility through migration of VMs between cloudlets [50].

Simulation Platform	Programming language	Availability	Documentation	Graphical support	Energy-aware	Cost-aware	Virtual machine support	Application models	Communication model	Migration support	Mobility/ Location-aware	Fog/Edge support	Last commit	Web page	Paper
CloudSim	Java	✓	✓		✓	✓	✓	✓	✓̸	✓			2016	[51]	[52]
CloudNetSim++	C++	✓		✓	✓	✓	✓	✓	✓	✓			2015	[53]	[48]
GreenCloud	C++/ Otcl	✓	✓		✓		✓	✓	✓	✓			2016	[54]	[47]
iCanCloud	C++	✓	✓	✓	✓	✓	✓	✓	✓				2015	[55]	[56]
CloudSched	Java	✓		✓	✓		✓						2015	[57]	[58]
CloudAnalyst*	Java	✓		✓		✓	✓	✓	✓̸	✓	✓		2009	[51]	[49]
DynamicCloudSim*	Java	✓			✓	✓	✓	✓	✓̸	✓			2017	[59]	[60]
CloudReports*	Java	✓		✓	✓	✓	✓	✓	✓̸	✓			2012	[61]	–
RealCloudSim*	Java	✓	✓̸	✓	✓	✓	✓	✓	✓	✓			2013	[62]	–
DCSim	Java	✓	✓̸		✓		✓	✓	✓̸	✓			2014	[63]	[64]
CloudSim Plus*	Java	✓	✓		✓	✓	✓	✓	✓	✓			2018	[65]	[66]
CloudSim Plus Automation*	Java	✓	✓		✓	✓	✓	✓	✓	✓			2018	[67]	–
DISSECT-CF	Java	✓	✓		✓		✓		✓̸	✓			2018	[68]	[69]
WorkflowSim*	Java	✓	✓		✓	✓	✓	✓	✓̸	✓			2015	[70]	[71]
Cloud2Sim*	Java	✓			✓	✓	✓	✓	✓̸	✓			2016	[72]	[73]
CloudSimDisk*	Java	✓			✓	✓	✓	✓	✓̸	✓			2015	[74]	[75]
RECAP	Java			–	✓	✓	✓	✓	✓	✓	✓	✓	–	[76]	–
iFogSim*	Java	✓	✓	✓	✓	✓	✓	✓	✓̸			✓	2016	[77]	[78]
MyiFogSim**	Java	✓		✓	✓	✓	✓	✓	✓̸	✓	✓	✓	2017	[79]	[50]
iFogSimWithData Placement**	Java	✓		✓	✓	✓	✓	✓	✓̸			✓	2018	[80]	[81]
EdgeCloudSim	Java	✓	✓			✓	✓	✓	✓		✓	✓	2018	[82]	[83]
YAFS	Python	✓	✓		✓̸			✓̸	✓̸			✓	2018	[84]	–
FogTorch	Java	✓					✓	✓	✓̸			✓	2016	[85]	[86]

Table 1: Comparison of fog and related computing paradigms simulators ('*' - extends CloudSim, '**' - extends both CloudSim and iFogSim, '✓̸' - limited, '–' - no information).

3 Algoritms

xxxx

4 Architecture

To this work is considered the typical architecture of fog computing, Fig. 2, where apart from the fixed cloudlets, there are some mobile fog nodes that can be used to support

individual applications or to support the fixes cloudlets, providing offloading support ..... tal como o paper do bus

5 Evaluation

The evaluation of the proposed architecture will be done xxxx

6 Schedule of Future Work

Future work is scheduled as follows:

- xxxx
- xxxx

7 Conclusion

xxxxx

References

1. D. Evans, "The internet of things: How the next evolution of the internet is changing everything," *CISCO white paper*, vol. 1, no. 2011, pp. 1–11, 2011.
2. "Cisco visual networking index: Global mobile data traffic forecast update, 2016–2021 white paper - cisco," https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper- c11-520862.html, (Accessed on 10/25/2018).
3. "What is cloud computing? - amazon web services," https://aws.amazon.com/what-is-cloud-computing/, (Accessed on 11/01/2018).
4. S. R. Ellis, K. Mania, B. D. Adelstein, and M. I. Hill, "Generalizeability of latency detection in a variety of virtual environments," in *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 48, no. 23. SAGE Publications Sage CA: Los Angeles, CA, 2004, pp. 2632–2636.
5. B. Taylor, Y. Abe, A. Dey, M. Satyanarayanan, D. Siewiorek, and A. Smailagic, "Virtual machines for remote computing: Measuring the user experience," *Carnegie Mellon University*, 2015.
6. T. Szigeti and C. Hattingh, *End-to-end qos network design*. Cisco press, 2005.
7. M. Satyanarayanan, "Cloudlets: at the leading edge of cloud-mobile convergence," in *Proceedings of the 9th international ACM Sigsoft conference on Quality of software architectures*. ACM, 2013, pp. 1–2.
8. F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*. ACM, 2012, pp. 13–16.
9. S. Davy, J. Famaey, J. Serrat-Fernandez, J. L. Gorricho, A. Miron, M. Dramitinos, P. M. Neves, S. Latré, and E. Goshen, "Challenges to support edge-as-a-service," *IEEE Communications Magazine*, vol. 52, no. 1, pp. 132–139, 2014.
10. T. Taleb and A. Ksentini, "Follow me cloud: interworking federated clouds and distributed mobile networks," *IEEE Network*, vol. 27, no. 5, pp. 12–19, 2013.
11. A. Yousefpour, C. Fung, T. Nguyen, K. Kadiyala, F. Jalali, A. Niakanlahiji, J. Kong, and J. P. Jue, "All one needs to know about fog computing and related edge computing paradigms: A complete survey," *arXiv preprint arXiv:1808.05283*, 2018.

12. "Cloudlet - wikipedia," https://en.wikipedia.org/wiki/Cloudlet, (Accessed on 10/31/2018).
13. M. Satyanarayanan, "Fundamental challenges in mobile computing," in *Proceedings of the fifteenth annual ACM symposium on Principles of distributed computing*. Acm, 1996, pp. 1–7.
14. M. Shiraz, A. Gani, R. H. Khokhar, and R. Buyya, "A review on distributed application processing frameworks in smart mobile devices for mobile cloud computing," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 3, pp. 1294–1313, 2013.
15. "Open edge computing," http://openedgecomputing.org/about.html, (Accessed on 10/24/2018).
16. P. by the OpenFog, C. Architecture, and W. Group, "Openfog reference architecture for fog computing," 0208.
17. "Etsi - multi-access edge computing," https://www.etsi.org/technologies-clusters/technologies/multi-access-edge-computing, (Accessed on 10/25/2018).
18. "Mobile edge computing vs. multi-access edge computing," https://www.sdxcentral.com/mec/definitions/mobile-edge-computing-vs-multi-access-edge-computing/, (Accessed on 10/25/2018).
19. T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On multi-access edge computing: A survey of the emerging 5g network edge cloud architecture and orchestration," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1657–1681, 2017.
20. M. Satyanarayanan, V. Bahl, R. Caceres, and N. Davies, "The case for vm-based cloudlets in mobile computing," *IEEE pervasive Computing*, 2009.
21. Y. Jararweh, L. Tawalbeh, F. Ababneh, and F. Dosari, "Resource efficient mobile computing using cloudlet infrastructure," in *Mobile Ad-hoc and Sensor Networks (MSN), 2013 IEEE Ninth International Conference on*. IEEE, 2013, pp. 373–377.
22. "Cisco pushes iot analytics to the extreme edge with mist computing - rethink," https://rethinkresearch.biz/articles/cisco-pushes-iot-analytics-extreme-edge-mist-computing-2/, (Accessed on 10/25/2018).
23. M. Iorga, L. Feldman, R. Barton, M. J. Martin, N. S. Goren, and C. Mahmoudi, "Fog computing conceptual model," Tech. Rep., 2018.
24. O. C. A. W. Group *et al.*, "Openfog reference architecture for fog computing," *OPFRA001*, vol. 20817, p. 162, 2017.
25. F. Bonomi, R. Milito, P. Natarajan, and J. Zhu, "Fog computing: A platform for internet of things and analytics," in *Big data and internet of things: A roadmap for smart environments*. Springer, 2014, pp. 169–186.
26. J. Gedeon, C. Meurisch, D. Bhat, M. Stein, L. Wang, and M. Mühlhäuser, "Router-based brokering for surrogate discovery in edge computing," in *Distributed Computing Systems Workshops (ICDCSW), 2017 IEEE 37th International Conference on*. IEEE, 2017, pp. 145–150.
27. E. Saurez, K. Hong, D. Lillethun, U. Ramachandran, and B. Ottenwälder, "Incremental deployment and migration of geo-distributed situation awareness applications in the fog," in *Proceedings of the 10th ACM International Conference on Distributed and Event-based Systems*. ACM, 2016, pp. 258–269.
28. J. Li, J. Jin, D. Yuan, and H. Zhang, "Virtual fog: A virtualization enabled fog computing framework for internet of things," *IEEE Internet Things J*, vol. 5, pp. 121–131, 2018.
29. N. K. Giang, M. Blackstock, R. Lea, and V. C. Leung, "Developing iot applications in the fog: a distributed dataflow approach," in *Internet of Things (IOT), 2015 5th International Conference on the*. IEEE, 2015, pp. 155–162.
30. T. Bahreini and D. Grosu, "Efficient placement of multi-component applications in edge computing systems," in *Proceedings of the Second ACM/IEEE Symposium on Edge Computing*. ACM, 2017, p. 5.
31. R. Bruschi, F. Davoli, P. Lago, and J. F. Pajo, "Move with me: Scalably keeping virtual objects close to users on the move," in *2018 IEEE International Conference on Communications (ICC)*. IEEE, 2018, pp. 1–6.
32. B. Ottenwälder, B. Koldehofe, K. Rothermel, and U. Ramachandran, "Migcep: operator migration for mobility driven distributed complex event processing," in *Proceedings of the 7th ACM international conference on Distributed event-based systems*. ACM, 2013, pp. 183–194.

33. R. Urgaonkar, S. Wang, T. He, M. Zafer, K. Chan, and K. K. Leung, "Dynamic service migration and workload scheduling in edge-clouds," *Performance Evaluation*, vol. 91, pp. 205–228, 2015.
34. W. Zhang, Y. Hu, Y. Zhang, and D. Raychaudhuri, "Segue: Quality of service aware edge cloud service migration," in *Cloud Computing Technology and Science (CloudCom), 2016 IEEE International Conference on*. IEEE, 2016, pp. 344–351.
35. W. Zhang, J. Chen, Y. Zhang, and D. Raychaudhuri, "Towards efficient edge cloud augmentation for virtual reality mmogs," in *Proceedings of the Second ACM/IEEE Symposium on Edge Computing*. ACM, 2017, p. 8.
36. S. Abdelwahab, S. Zhang, A. Greenacre, K. Ovesen, K. Bergman, and B. Hamdaoui, "When clones flock near the fog," *IEEE Internet of Things Journal*, 2018.
37. X. Sun and N. Ansari, "Primal: Profit maximization avatar placement for mobile edge computing," in *Communications (ICC), 2016 IEEE International Conference on*. IEEE, 2016, pp. 1–6.
38. D. Ye, M. Wu, S. Tang, and R. Yu, "Scalable fog computing with service offloading in bus networks," in *Cyber Security and Cloud Computing (CSCloud), 2016 IEEE 3rd International Conference on*. IEEE, 2016, pp. 247–251.
39. L. F. Bittencourt, J. Diaz-Montes, R. Buyya, O. F. Rana, and M. Parashar, "Mobility-aware application scheduling in fog computing," *IEEE Cloud Computing*, vol. 4, no. 2, pp. 26–35, 2017.
40. M. M. Lopes, W. A. Higashino, M. A. Capretz, and L. F. Bittencourt, "Myifogsim: A simulator for virtual machine migration in fog computing," in *Companion Proceedings of the10th International Conference on Utility and Cloud Computing*. ACM, 2017, pp. 47–52.
41. X. Sun and N. Ansari, "Avaptive avatar handoff in the cloudlet network," *IEEE Transactions on Cloud Computing*, no. 1, pp. 1–1, 2017.
42. W. Bao, D. Yuan, Z. Yang, S. Wang, W. Li, B. B. Zhou, and A. Y. Zomaya, "Follow me fog: Toward seamless handover timing schemes in a fog computing environment," *IEEE Communications Magazine*, vol. 55, no. 11, pp. 72–78, 2017.
43. L. Ma, S. Yi, and Q. Li, "Efficient service handoff across edge servers via docker container migration," in *Proceedings of the Second ACM/IEEE Symposium on Edge Computing*. ACM, 2017, p. 11.
44. Y. Sun, S. Zhou, and J. Xu, "Emm: Energy-aware mobility management for mobile edge computing in ultra dense networks," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 11, pp. 2637–2646, 2017.
45. Y. Bi, G. Han, C. Lin, Q. Deng, L. Guo, and F. Li, "Mobility support for fog computing: An sdn approach," *IEEE Communications Magazine*, vol. 56, no. 5, pp. 53–59, 2018.
46. I. Farris, T. Taleb, M. Bagaa, and H. Flick, "Optimizing service replication for mobile delay-sensitive applications in 5g edge network," in *Communications (ICC), 2017 IEEE International Conference on*. IEEE, 2017, pp. 1–6.
47. D. Kliazovich, P. Bouvry, and S. U. Khan, "Greencloud: a packet-level simulator of energy-aware cloud computing data centers," *The Journal of Supercomputing*, vol. 62, no. 3, pp. 1263–1283, 2012.
48. A. W. Malik, K. Bilal, S. Malik, Z. Anwar, K. Aziz, D. Kliazovich, N. Ghani, S. U. Khan, and R. Buyya, "Cloudnetsim++: a gui based framework for modeling and simulation of data centers in omnet++," *IEEE Transactions on Services Computing*, no. 4, pp. 506–519, 2017.
49. B. Wickremasinghe, R. N. Calheiros, and R. Buyya, "Cloudanalyst: A cloudsim-based visual modeller for analysing cloud computing environments and applications," in *Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on*. IEEE, 2010, pp. 446–452.
50. M. M. Lopes, W. A. Higashino, M. A. Capretz, and L. F. Bittencourt, "Myifogsim: A simulator for virtual machine migration in fog computing," in *Companion Proceedings of the10th International Conference on Utility and Cloud Computing*. ACM, 2017, pp. 47–52.
51. "The clouds lab: Flagship projects - gridbus and cloudbus," http://www.cloudbus.org/cloudsim/, (Accessed on 11/15/2018).

52. R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, "Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and experience*, vol. 41, no. 1, pp. 23–50, 2011.
53. "cloudnetsim.seecs.edu.pk," http://cloudnetsim.seecs.edu.pk/, (Accessed on 11/15/2018).
54. "Greencloud - the green cloud simulator," https://greencloud.gforge.uni.lu/, (Accessed on 11/15/2018).
55. "Web site," https://www.arcos.inf.uc3m.es/old/icancloud/Home.html, (Accessed on 11/15/2018).
56. A. Núñez, J. L. Vázquez-Poletti, A. C. Caminero, G. G. Castañé, J. Carretero, and I. M. Llorente, "icancloud: A flexible and scalable cloud infrastructure simulator," *Journal of Grid Computing*, vol. 10, no. 1, pp. 185–209, 2012.
57. "Cloudsched download — sourceforge.net," https://sourceforge.net/projects/cloudsched/, (Accessed on 11/15/2018).
58. W. Tian, Y. Zhao, M. Xu, Y. Zhong, and X. Sun, "A toolkit for modeling and simulation of real-time virtual machine allocation in a cloud data center," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 1, pp. 153–161, 2015.
59. "marcbux/dynamiccloudsim: Automatically exported from code.google.com/p/dynamiccloudsim," https://github.com/marcbux/dynamiccloudsim, (Accessed on 11/15/2018).
60. M. Bux and U. Leser, "Dynamiccloudsim: Simulating heterogeneity in computational clouds," *Future Generation Computer Systems*, vol. 46, pp. 85–99, 2015.
61. "thiagotts/cloudreports: An extensible simulation tool for energy-aware cloud computing environments," https://github.com/thiagotts/CloudReports, (Accessed on 11/15/2018).
62. "Realcloudsim download — sourceforge.net," https://sourceforge.net/projects/realcloudsim/, (Accessed on 11/15/2018).
63. "digs-uwo/dcsim: Dcsim: A data centre simulation tool for evaluating dynamic virtualized resource management," https://github.com/digs-uwo/dcsim, (Accessed on 11/15/2018).
64. M. Tighe, G. Keller, M. Bauer, and H. Lutfiyya, "Dcsim: A data centre simulation tool for evaluating dynamic virtualized resource management," in *Network and service management (cnsm), 2012 8th international conference and 2012 workshop on systems virtualiztion management (svm)*. IEEE, 2012, pp. 385–392.
65. "Cloudsim plus — a modern, full-featured, highly extensible and easier-to-use java 8 framework for modeling and simulation of cloud computing infrastructures and services," http://cloudsimplus.org/, (Accessed on 11/15/2018).
66. M. C. Silva Filho, R. L. Oliveira, C. C. Monteiro, P. R. Inácio, and M. M. Freire, "Cloudsim plus: a cloud computing simulation framework pursuing software engineering principles for improved modularity, extensibility and correctness," in *Integrated Network and Service Management (IM), 2017 IFIP/IEEE Symposium on*. IEEE, 2017, pp. 400–406.
67. "manoelcampos/cloudsim-plus-automation: Human readable scenario specification for automated creation of simulations on cloudsim plus and cloudsim," https://github.com/manoelcampos/cloudsim-plus-automation, (Accessed on 11/15/2018).
68. "kecskemeti/dissect-cf: Discrete event based energy consumption simulator for clouds and federations," https://github.com/kecskemeti/dissect-cf, (Accessed on 11/15/2018).
69. G. Kecskemeti, "Dissect-cf: a simulator to foster energy-aware scheduling in infrastructure clouds," *Simulation Modelling Practice and Theory*, vol. 58, pp. 188–218, 2015.
70. "Workflowsim/workflowsim-1.0: Wiki pages," https://github.com/WorkflowSim/WorkflowSim-1.0, (Accessed on 11/15/2018).
71. W. Chen and E. Deelman, "Workflowsim: A toolkit for simulating scientific workflows in distributed environments," in *E-science (e-science), 2012 IEEE 8th International Conference on*. IEEE, 2012, pp. 1–8.
72. "Cloud2sim download — sourceforge.net," https://sourceforge.net/projects/cloud2sim/, (Accessed on 11/15/2018).
73. P. Kathiravelu and L. Veiga, "An adaptive distributed simulator for cloud and mapreduce algorithms and architectures," in *Utility and Cloud Computing (UCC), 2014 IEEE/ACM 7th International Conference on*. IEEE, 2014, pp. 79–88.

74. "Udacity2048/cloudsimdisk: A new storage modeling for cloudsim simulator." https://github.com/Udacity2048/CloudSimDisk, (Accessed on 11/15/2018).
75. B. Louis, K. Mitra, S. Saguna, and C. Åhlund, "Cloudsimdisk: Energy-aware storage simulation in cloudsim," in *Utility and Cloud Computing (UCC), 2015 IEEE/ACM 8th International Conference on*. IEEE, 2015, pp. 11–15.
76. "Recap — reliable capacity provisioning and enhanced remediation for distributed cloud applications," https://recap-project.eu/, (Accessed on 11/15/2018).
77. "Cloudslab/ifogsim: The ifogsim toolkit for modeling and simulation of resource management techniques in internet of things, edge and fog computing environments," https://github.com/Cloudslab/iFogSim, (Accessed on 11/15/2018).
78. H. Gupta, A. Vahid Dastjerdi, S. K. Ghosh, and R. Buyya, "ifogsim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments," *Software: Practice and Experience*, vol. 47, no. 9, pp. 1275–1296, 2017.
79. "marciocomp/myifogsim: Myifogsim is a simulator for fog computing concerns. it extends the ifogsim simulator to support virtual machine migration policies for mobile users," https://github.com/marciocomp/myifogsim, (Accessed on 11/15/2018).
80. "medislam/ifogsimwithdataplacement," https://github.com/medislam/iFogSimWithDataPlacement, (Accessed on 11/15/2018).
81. M. I. Naas, J. Boukhobza, P. R. Parvedy, and L. Lemarchand, "An extension to ifogsim to enable the design of data placement strategies," in *Fog and Edge Computing (ICFEC), 2018 IEEE 2nd International Conference on*. IEEE, 2018, pp. 1–8.
82. "Cagataysonmez/edgecloudsim: Edgecloudsim: An environment for performance evaluation of edge computing systems," https://github.com/CagataySonmez/EdgeCloudSim, (Accessed on 11/15/2018).
83. C. Sonmez, A. Ozgovde, and C. Ersoy, "Edgecloudsim: An environment for performance evaluation of edge computing systems," in *Fog and Mobile Edge Computing (FMEC), 2017 Second International Conference on*. IEEE, 2017, pp. 39–44.
84. "yafs pypi," https://pypi.org/project/yafs/, (Accessed on 11/15/2018).
85. "di-unipi-socc/fogtorch," https://github.com/di-unipi-socc/FogTorch, (Accessed on 11/15/2018).
86. A. Brogi and S. Forti, "Qos-aware deployment of iot applications through the fog," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1185–1192, 2017.