DOI: 10.14529/jsfi210302

Fog Computing State of the Art: Concept and Classification of Platforms to Support Distributed Computing Systems

Alexandra A. Kirsanova[1], *Gleb I. Radchenko*[1], *Andrei N. Tchernykh*[1,2,3]

As the Internet of Things (IoT) becomes a part of our daily life, there is a rapid growth in the connected devices. A well-established approach based on cloud computing technologies cannot provide the necessary quality of service in such an environment, particularly in terms of reducing data latency. Today, fog computing technology is seen as a novel approach for processing large amounts of critical and time-sensitive data. This article reviews cloud computing technology and analyzes the prerequisites for the evolution of this approach and the emergence of the concept of fog computing. As part of an overview of the critical features of fog computing, we analyze the frequent confusion of the concepts of fog and edge computing. We provide an overview of fog computing technologies: virtualization, containerization, orchestration, scalability, parallel computing environments, as well as systematic analysis of the most popular platforms that support fog computing. As a result of the analysis, we offer two approaches to classification of the fog computing platforms: by the principle of openness/closure of components and by the three-level classification based on the provided platform functionality (Deploy-, Platform- and Ecosystem as a Service).

Keywords: big data processing, fog computing, scheduling, cloud computing, edge computing, Internet of Things.

Introduction

Data is a major commodity today. Having more data and the ability to intelligently analyze it effectively creates significant value for data-managed enterprises [40]. According to the International Data Corporation (IDC), the amount of digital data generated in 202 exceeded 59 zettabytes (ZB) of data [5]. Cisco estimated that there will be about 50 billion connected devices in 2020 [27]. These connected devices form the Internet of Things (IoT) and generate a vast amount of data in real-time. Modern mobile networks are already being designed considering the loads that arise in the transmission and processing of such astronomical volumes of data.

Within the cloud computing concept, most of the data that requires storage, analysis, and decision making is sent to data centers in the cloud [74]. As the data volume increases, moving information between an IoT device and the cloud may be inefficient or even impossible in some cases due to bandwidth limitations or latency requirements. As time-sensitive applications (such as patient monitoring, autopilot vehicles, etc.) become more common, the remote cloud will not be able to meet the need for ultra-reliable communications with minimal delay [99]. Moreover, some applications may not be able to send data to the cloud because of privacy issues.

To solve the challenges of applications that require high network bandwidth, access to geographically distributed data sources, ultra-low latency, and localized data processing, there is a specific need for a computing paradigm that provides a one-size-fits-all approach to the organization of computing, both in the cloud and in computing nodes closer to connected devices. The concept of Fog computing has been proposed by industry and academia to bridge the gap

[1]South Ural State University, Chelyabinsk, Russia
[2]CICESE Research Center Ensenada, Mexico
[3]Ivannikov Institute for System Programming of the RAS, Russia

between the cloud and IoT devices by providing computing capabilities, storage, networking, and data management at network nodes closely located to IoT devices [15, 68]. The research community has proposed several computing paradigms to address these problems, such as edge computing, fog computing, and dew computing. A common feature of these concepts is the use of distributed heterogeneous systems that provide highly scalable clusters of computing nodes located closely (either networked or geographically) to data sources. In this review, we provide an analysis of the most popular platforms that support fog computing solutions. Based on this analysis, we propose two approaches to classify fog computing platforms: by the principle of openness/closure of components and by the three-tier classification based on the provided platform functionality (Deploy-, Platform- and Ecosystem as a Service).

The article is organized as follows. Section 1 discusses cloud computing as the basis for new computing concepts, prerequisites for the emergence, and key characteristics of cloud computing. Section 2 is devoted to fog and edge computing, their origins, definition, and critical characteristics. Section 3 discusses technologies that support fog computing, including virtualization, orchestration, security, and computation scalability issues. Section 4 provides an overview of fog computing platforms: private, public, open-source, and proposes a classification of fog platforms. In section 5 we focus on the current challenges faced by the fog computing researchers. In conclusion, we summarize the results obtained in the context of this study and indicate directions for further research.

1. Cloud Computing as a Basis for New Computational Concepts

1.1. The Prerequisites for Cloud Computing

The utility computing concept, originating in the 1960s, is considered to be the earliest ancestor of cloud technologies [31, 93]. This concept was not generally adopted until the 90s due to the technical constraints of the deployment and use of this architecture [13, 17, 41, 58, 60, 93]. Improvements in network technology and data transfer rates in the mid-'90s led to a new round of research in utility computing in the framework of the grid computing concept [33, 41, 58]. These shortcomings have led to further evolutionary development and the emergence of cloud computing, which often uses the grid computing model to expand computing resources [55].

1.2. Key Features of Cloud Computing

Today, cloud computing systems have become widely used for Big Data processing, providing access to a wide variety of computing resources and a greater distribution between multiclouds [73]. This trend has been strengthened by the rapid development of the Internet of Things (IoT) concept. Virtualization via virtual machines and containers is a traditional way of organization of cloud computing infrastructure. Containerization technology provides a lightweight virtual runtime environment. In addition to the advantages of traditional virtual machines in terms of size and flexibility, containers are particularly important for integration tasks for PaaS solutions, such as application packaging and service orchestration.

The National Institute of Standards and Technology (NIST) published a definition of cloud computing, its main characteristics, and its deployment and maintenance models in 2011. Cloud computing has been defined as a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.

The NIST model comprises five essential characteristics, three service models, and four deployment models for clouds [63]. The following key cloud deployment models can be identified as: private cloud, public cloud, and hybrid cloud [26, 94].

A private cloud is deployed within a single organization, is available only to internal users, and does not share its resources outside the organization. The public cloud is developed by third parties and provides the resources to external users under the terms of the contract on the right of use. A hybrid cloud combines two types of deployment described above, which allows to build a balance between private and public computing [26].

Private clouds are commonly deployed as close to the end-user of the cloud as possible. That reduces the response time of the computing platform and increases the speed of data transfer between the nodes of the system. However, a private cloud is tightly interconnected with the computing needs of its owner. Not every organization has enough resources to maintain its private cloud, which must meet the requirements for availability, reliability, and the law's requirements in the country where the cloud is located [44, 78].

On the other hand, public cloud users often lack direct control over the underlying computing infrastructure. This can lead to several problems, including uncontrolled access by third parties to the private data hosted in a public cloud; blocking user servers that can be deployed on the same subnet with hosts banned in a particular country; the uncertainty of the quality of cloud resources as they are deployed on servers shared with third parties [44]. It is also challenging to ensure a change of cloud provider, as it is necessary to solve the problem of migration and conversion of data and computing services.

These features of each type of deployment are the reason why cloud providers that provide clouds to private organizations often support the ability to create hybrid clouds [84], which can be configured to a particular mode of operation, depending on the customer's requirements. This approach addresses data latency, security, and migration issues while maintaining the flexibility to customize computing resources for each task.

1.3. Preconditions for New Computing Concepts

Despite all the significant advantages guaranteed by public cloud platforms, problems that such approaches cannot effectively solve have emerged over the last five years. Thus, a large number of users of "smart" systems such as "smart home", "smart enterprise", "smart city" and other IoT solutions cannot always be satisfied with the quality of services provided by cloud solutions, in particular, due to the increase in the amount of data sent between the user/device and the cloud [46].

The emergence of the smart systems approach, populated with a variety of Internet-connected sensors and actuators, led to a revision of the architectural concept of data collection and analysis systems. The Internet of Things concept requires new approaches to storage solutions, fast data processing, and the ability to respond quickly to changes in the state of end devices [69, 70, 98]. Also, the spread of mobile devices as the main platforms for client applications makes it difficult to transfer and process large amounts of data without causing problems with response delays due to the constant movement of mobile devices.

As the amount of data sent between IoT devices, clients, and the cloud increases, problems associated with increased response time due to physical bandwidth limitations appear [60]. On the other hand, there are response time-sensitive applications and devices such as life support systems, autopilots, drones and others. Under these conditions, a remote centralized cloud has become unable to meet the ultra-low latency requirements [98]. Also, data transmission through multiple gateways and subnets raises the issue of sensitive data transmission [51].

In response to these problems, private enterprises and the academic community have raised the need to develop a computing paradigm that meets new concepts such as IoT [15, 61, 70]. This paradigm had to fill the gap between the cloud and the end devices, providing computing, storage, and data transfer in intermediate network nodes closest to the end devices. Several paradigms have been developed and applied to solve this problem, including fog and edge computing [23]. Each of these paradigms has its specific features, but all of them derive from a common principle – reducing time delays in data processing and transmission by moving computing tasks closer to the final device.

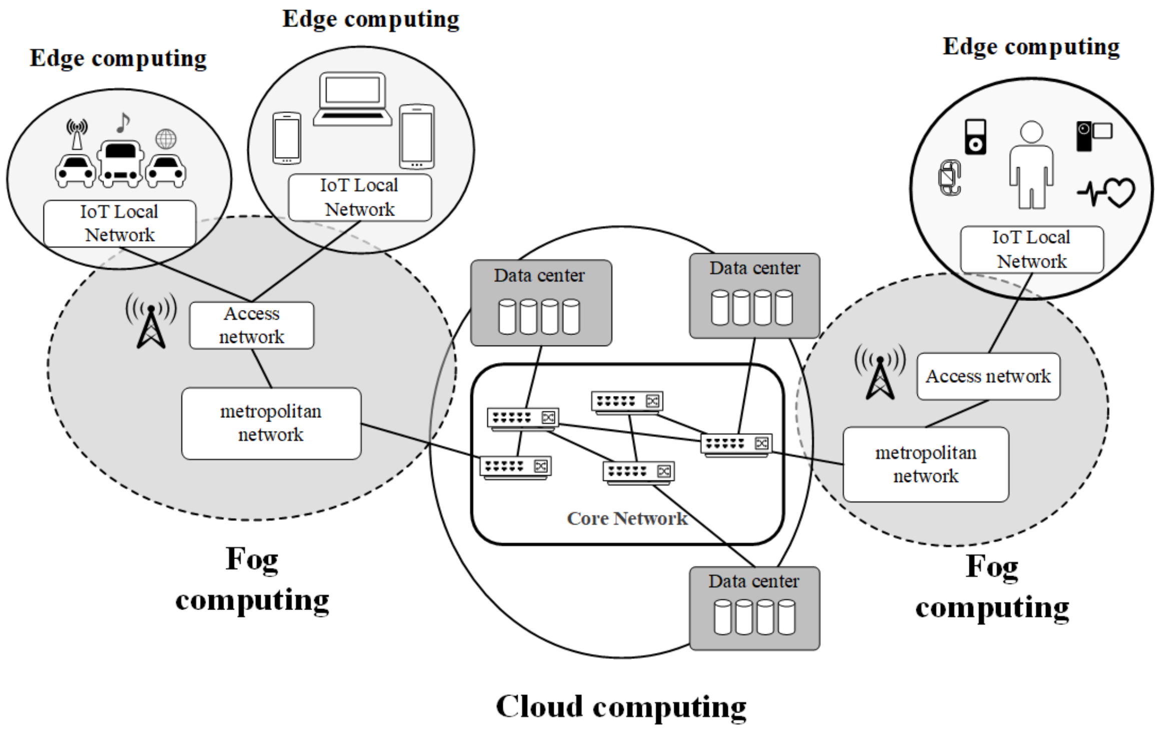

Figure 1. Comparison of the infrastructure of fog computing and its related computing paradigms from the networking perspective [93]

Figure 1 shows a diagram of the relative distribution of computational resources defined by edge, fog and cloud computing concepts. Cloud computing is a separate data center (DC) or a network of data centers located far from the user but providing high computing capabilities. On the other hand, edge computing is located right at the edge of the computing system and provides small computing capabilities, but near the consumer of those resources. Fog computing is located between the edge of the network and the cloud data center, providing significant computing resources close to the end-user, which, on the other hand, is not comparable to the total amount of cloud computing resources but can be customized and scale depending on the objectives of the end-user. This article considers Fog computing as a more general concept that includes the edge computing paradigm [45].

2. Fog and Edge Computing

2.1. History and Definition

In 1961 (see Tab. 1), John McCarthy spoke at the MIT Centennial: "If computers of the kind I have advocated become the computers of the future, then computing may someday be organized as a public utility just as the telephone system is a public utility... The computer utility could become the basis of a new and important industry." [77] His concept was the basis for the idea of Douglas Parkhill [41, 43, 77, 89] to create a grid computing paradigm that was described later in 1966 and was a set of computers connected over a grid that take the computing decisions collectively.

The fog computing approach was one of the first technologies to solve the latency issues of cloud computing. The "Fog Computing" term was first proposed by CISCO in 2012 [42] and had been described as "a highly virtualized platform that provides compute storage, and networking services between end devices and traditional Cloud Computing Data Centers, typically, but not exclusively located at the edge of the network" [15]. The OpenFog group was established in 2015 to develop standards in the field of fog computing. It included companies and academic organizations such as Cisco, Dell, Intel, Microsoft Corp, and Princeton University. On December 18, 2018, the OpenFog consortium became part of The Industrial Internet Consortium [50].

Table 1. Fog computing timeline

1961	1990's	2012	2015	2018
John McCarthy. Utility computing Definition [76]	**Ian Foster et. al**. Definition of the grid computing [33]	**Flavio Bonomi et. al**. CISCO proposed the definition of the cloud computing [15] **Mell Peter**. The NIST published the definition of the cloud computing [63]	**The OpenFog group** was established [68]	**Machaela Iorga et. al**. The NIST published the definition of the fog computing [51] **Mahmoudi Charid**. The formal definition of the edge computing was published [62]

In 2018, the National Institute of Standards and Technology of the United States had formulated an official definition of the fog computing term: "Fog computing is a layered model for enabling ubiquitous access to a shared continuum of scalable computing resources. The model facilitates the deployment of distributed, latency-aware applications and services, and consists of fog nodes (physical or virtual), residing between smart end-devices and centralized (cloud) services. The fog nodes are context-aware and support a common data management and communication system. They can be organized in clusters – either vertically (to support isolation), horizontally (to support federation), or relative to fog nodes latency-distance to the smart end-devices. Fog computing minimizes the request-response time from/to supported applications, and provides, for the end-devices, local computing resources and, when needed, network connectivity to centralized services" [51].

Bridging the gap between the cloud and end devices through computing, storage, and data management not only in the cloud but also in intermediate nodes [59] has expanded the scope of fog computing, which allowed its application in new tasks such as IoT, smart vehicles [47], smart cities [22], health care [37], smart delivery (including the use of drones) [92], video surveillance, etc. [100]. These systems benefit significantly from Big Data processing [76], allowing them to extract new knowledge and decision-making information from the data streams generated by clusters of IoT devices. Fog computing supports this challenge by enabling distributed computing resources for lightweight data processing tasks, including filtering and preprocessing data before sending it to the cloud. But the geographical distribution, heterogeneity of computing nodes, and high instability of network communications at the edge level lead to the need to solve complex problems associated with monitoring, scheduling, and ensuring the necessary quality of service of such services.

2.2. Key Characteristics of Fog Computing

Due to the late separation of the fog and edge computing concepts, many companies introduced their characteristics [9] and definitions for fog and edge computing, often combining them into one [59]. Table 2 presents the key characteristics that different authors distinguished for fog and edge computing.

In 2017, the OpenFog Consortium released a reference architecture for fog computing, which is based on eight basic principles: security, scalability, openness, autonomy, RAS (reliability, availability, and serviceability), agility, hierarchy, and programmability [68].

Table 2. Characteristics of Fog Computing [65]

Method	Properties									
	Highly virtualized	**Generally used for IoT**	**Extends the cloud**	**Not exclusively located at the edge**	**Resides at network ends**	**Fog device consists of processing, storage, and network connectivity**	**Run in a sandboxed environment**	**Leasing part of users devices and provide an incentive**	**Fog and Edge computing as similar**	**Can be deployed anywhere**
Bonomi et al. [15]	+	+		+						
Cisco Systems [21]		+	+			+				+
Vaquero and Rodero-Merino [36]						+	+	+		
IBM [49]			+	+	+				+	
Synthesis [65]	+		+	+	+					

In [45] and [11], the following key characteristics of fog computing are highlighted.

- *Contextual location awareness and low latency.* Fog computing offers the lowest-possible latency due to the fog nodes awareness of their logical location in the context of the entire system and of the latency costs for communicating with other nodes.
- *Geographical distribution.* In sharp contrast to the more centralized cloud, the services and applications targeted by fog computing demand widely but geographically identifiable, distributed deployments.

- *Heterogeneity.* Fog computing supports the collection and processing of data of different form factors acquired through multiple types of network communication capabilities
- *Interoperability and federation.* Seamless support of certain services (real-time streaming services is a good example) requires the cooperation of different providers. Hence, fog computing components must be able to interoperate, and services must be federated across domains.
- *Real-time interactions.* Fog computing applications involve real-time interactions rather than batch processing.
- *Scalability and agility of federated, fog-node clusters.* Fog computing is adaptive, at cluster or cluster-of-clusters level, supporting elastic compute, resource pooling, data-load changes, and network condition variations, to list a few of the supported adaptive functions.
- *Cognition.* Cognition is responsiveness to client-centric objectives. Fog-based data access and analytics give a better alert about customer requirements, best position handling for transmitting, storing, and controlling functions throughout the cloud to the IoT continuum. Applications, due to proximity, at end devices provide a better conscious and responsive reproduced customer requirement relation [97].
- *Support for Mobility.* Mobility support is a vital fog computational advantage that can enable direct communication between mobile devices using SDN protocols (i.e., CISCO Locator/ID Separation Protocol) that decouples host identity from location identity with a dispersed indexing system [102].
- *Large Scale Sensor Network.* The fog has a feature applicable when an environment monitoring system, in near smart grid applications, inherently extends its monitoring systems caused by hierarchical computing and storage resource requirements.
- *Widespread Wireless Access.* In this scenario, wireless access protocols (WAP) and cellular mobile gateways can act as typical examples of fog node proximity to the end-users.
- *Interoperable Technology.* Fog components must work in an interoperating environment to guarantee support for a wide range of services like data streaming and real-time processing for best data analyses and predictive decisions.

2.3. Fog and Edge Computing Concepts Definitions

Some sources refer to fog computing as edge computing, relying on the critical technology feature that data collection and analysis is not organized in a centralized cloud, but as close to the end device as possible, "at the edge of the network" [15, 34, 46, 51].

However, [98] indicates that although fog and edge computing move computation and data storage closer to the network edge, these paradigms are not identical. Within the Fog Computing paradigm, fog nodes are located at the edge of the local network, often they are deployed based on routers and wireless access points (if these devices support the required technologies for deployment of the fog node) [92]. In contrast to fog computing, edge computing is deployed even "closer" to the end devices, already inside the local network itself on the intermediate access points. Sometimes the end devices themselves can act as edge computing nodes. Smartphones, tablets, and other computing devices with sufficient computing capabilities and support for the deployment of computing nodes can handle edge computing tasks [88]. However, this also limits their computational power, and therefore there are some limitations in their application scope.

So, edge computing is used to solve such tasks as video surveillance, video caching, and traffic control [98].

The OpenFog Consortium claims that edge computing is often erroneously referred to as fog computing and determine that the main difference is that fog computing is the overall architecture of distributing resources across the network, whereas edge computing is specifically focused on executing compute processes close to end-users outside the core of the network [62]. In [20], the authors note on fog and edge computing that "fog is inclusive of cloud, core, metro, edge, clients, and things" and "the fog seeks to realize a seamless continuum of computing services from the cloud to the things rather than treating the network edges as isolated computing platforms".

Thus, the term "edge computing" is mainly used in the telecommunications industry and usually refers to 4G/5G, RAN (Radio Access Network), and ISP (Internet Service Provider) base stations [20, 56]. However, this term has recently been used in the subject area of IoT [35, 56, 75] concerning the local network where sensors and IoT devices are located. In other words, "edge computing" is located within the first of the IoT device of the transit section of the network, for example, at WiFi access points or gateways.

2.4. Classification of Fog Computing Applications

Fog computing enables new applications, especially those with strict latency constraints and those involving mobility. These new applications have heterogeneous QoS requirements and demand Fog management mechanisms to cope efficiently with that heterogeneity. Thus, resource management in Fog computing is quite challenging, calling for integrated mechanisms capable of dynamically adapting the allocation of resources. The very first step in resource management is to separate the incoming flow of requests into Classes of Service (CoS) according to their QoS requirements. The mapping of applications into a set of classes of service is the first step in creating a resource management system capable of coping with the heterogeneity of Fog applications. The authors of [38] proposed the following critical classes of fog computing applications:

- *Mission-critical.* Applications in which a component failure would cause a significant increase in the safety risk for people and the environment. Those are healthcare systems, criminal justice, drone operations, industrial control, financial transactions, military, and emergency operations. Those applications should implement distribution features to ensure duplication of functionality.
- *Real-time.* The speed of response in these applications is critical since data are processed at the same time they are generated but can tolerate a certain amount of data loss (online gaming, virtual and augmented reality applications).
- *Interactive.* Responsiveness is critical; the time between when the user requests and actions is less than a few seconds. Those are interactive television, web browsing, database retrieval, server access applications.
- *Conversational.* Characterized by being delay-sensitive but loss-tolerant with slight delays (about 100–200 ms). E.g., video and Voice-over-IP (VoIP) applications where losses cause occasional glitches in audio or video playback.
- *Streaming* class applications are accessed by users on-demand and must guarantee interactivity and continuous playout. The network must provide each stream with an average throughput that is larger than the content consumption rate. In such a case, data should

be located as close to the end-user as possible, and new nodes should easily be created and removed from the environment.

- *CPU-bound.* Involves complex processing models, such as those in decision making, which may demand hours, days, or even months of processing. Face recognition, animation rendering, speech processing, and distributed camera networks are examples of this applications class.
- *Best-effort.* For these applications, long delays are annoying but not particularly harmful; however, the completeness and integrity of the transferred data are of paramount importance. Some examples of the Best-Effort class are e-mail downloads, chats, SMS delivery, FTP, P2P file sharing.

3. Technologies that Support Fog and Edge Computing

3.1. Virtualization

The key technology that supports cloud and fog computing is virtualization [87], which allows to use the resources of one physical machine by several logical virtual machines (VMs) at the level of Hardware Abstraction Layer (HAL). Virtualization technology uses a hypervisor – a software layer that provides the operation of virtual machines based on hardware resources. A machine with a hypervisor is called a host machine. A virtual machine running on the host machine is called a guest machine, on which in turn the guest operating systems (OS) can be installed. This type of virtualization is called hypervisor-based virtualization.

There is also container-based virtualization [25], representing a packaged, standalone, deployable set of application components that can also include middleware and business logic in binary files and libraries to run applications.

Authors of [73] present a comparative analysis of both types of virtualization, based on which we can highlight some of the advantages of container-based virtualization.

- *Hardware costs.* Virtualization via containers decreases hardware costs by enabling consolidation. It enables concurrent software to take advantage of the true concurrency provided by a multicore hardware architecture.
- *Scalability.* A single container engine can efficiently manage large numbers of containers, enabling additional containers to be created as needed.
- *Spatial isolation.* Containers support lightweight spatial isolation by providing each container with its resources (e.g., core processing unit, memory, and network access) and container-specific namespaces.
- *Storage.* Compared with virtual machines, containers are lightweight concerning storage size. The applications within containers share both binaries and libraries.
- *Real-time applications.* Containers provide more consistent timing than virtual machines, although this advantage is lost when using hybrid virtualization which uses both types of virtualization (hypervisor and container-based).
- *Portability.* Containers support portability from development to production environments, especially for cloud-based applications.

Thus, two main virtualization technologies are currently used to support fog computing [53]: hypervisor-based and container-based. Cloud computing mainly uses hypervisor-based virtualization to share limited hardware resources among several virtual machines. Fog computing that commonly hosted on low-performance hardware prefers container-based virtualization to create node instances on new hardware devices. That is why container-based virtualization is becoming more and more widespread in fog computing. Due to lower hardware performance requirements to ensure the deployment of computing nodes, intermediate devices may not have high computing power. This is especially relevant for edge computing nodes because they are not even run on the IoT devices themselves [71] but on intermediate access points closest to the IoT devices.

3.2. Fog Computing Orchestration

With containerization evolving as one of the technologies to support fog computing, the challenge arose to manage the computational load to ensure efficient use of geographically dispersed resources [52]. Fog computing implementation requires a different level of computing resource management compared to the cloud, for example [91].

The first complex task that arises when working with fog computing, as opposed to cloud computing, is managing the distribution of computational load (orchestration) between nodes of the fog [56, 58] by placing the fog services on them, as well as orchestration of these services, i.e. ensuring efficient collaboration of computational services for solving tasks assigned to the fog environment. Authors of [95] formulate that orchestration provides the centralized arrangement of the resource pool, mapping applications with specific requests and providing an automated workflow to physical resources (deployment and scheduling); workload execution management with runtime QoS control; and time-efficient directive generation to manipulate specific objects.

Let us consider the key tasks to be solved by the Fog Orchestrator [24, 91].

- *Scheduling.* It is necessary to consider how to exploit the collaboration between nodes to offload applications (which were not used for a long time and should be deleted to save resources) efficiently in Fog environments. In general, the processing nodes should be managed by a resource broker in the Orchestrator to perform smart scheduling of the resource, considering the applications workflows.
- *Path computation's* main objectives are: maintaining end-to-end connectivity, adapting to dynamic topologies, maximizing network and application traffic performance and providing network resilience.
- *Discovery and allocation* of the physical and virtual devices in the Fog, as well as the resources associated with them.
- *Interoperability* is the ability that distributed system elements are able to interact with each other. Several factors influence the interoperability of a system, such as the heterogeneity of its elements.
- *Latency.* One of the characteristics of Fog environments is that they provide low levels of latency. This allows the deployment of a different kind of services with real-time and low latency restrictions that are not necessarily fit for the cloud; but also requires a new set of mechanisms that guarantee that these low latency levels are met.
- *Resilience.* To guarantee a smooth work of the complex and diverse environment where the IoT acts from the resilience perspective, an Orchestrator should be in charge of intelligent migration and instantiation of resources and services providing a global view on the status of the IoT.
- *Prediction and optimization.* Proper management of resources and services in an IoT environment, where these are geographically distributed, generating multi-dimensional data in enormous quantities, is only possible if the orchestration process takes into consideration

prediction and optimization mechanisms of all overlapping and interconnected layers in the IoT.
- *Security and privacy.* From the privacy perspective, the main challenge lies in preserving the end-user privacy since the Fog nodes are deployed near them, collecting sensitive data concerning identity and usage patterns. Regarding security, a significant challenge is how to deal with the massively distributed approach of the Fog to guarantee the proper authentication mechanisms and avoid massive distributed attacks.
- *Authentication, access, and account.* To perform activities related to application life cycle management (i.e. deployment, migration, application of policies), the Orchestrator interacts with the fog nodes in the environment.

Optimization of various metrics (latency, bandwidth, energy consumption etc.) plays a vital role in fog computing orchestration. The following key tasks related to the distribution of tasks and data by the level of fog computing are currently being identified [14]:
- Offloading computing tasks from end devices to fog nodes and cloud.
- Scheduling of tasks within a fog node.
- Clustering of fog nodes: how to determine the size of a cluster of fog nodes to handle the relevant requests.
- Migration of data/applications between fog nodes.
- Geographical distribution of physical resources (before operation).
- Distributing applications/data among fog nodes and cloud.

3.3. Fog Computing and Security Issues

Due to the significant degree of decentralization of the computing process, security in fog computing differs in some critical aspects from mechanisms used, for example, in cloud computing. The design of a secure fog architecture must take into account the security features of each layer of the computing architecture, including the features of lightweight wireless data transfer at the sensing/edge layer; data transfer over middleware mesh networks; preprocessing of data using clusters of fog nodes on the application level; possible data transfer over the WAN for processing in the public cloud [10, 72].

Each of these layers has its security issues and vulnerabilities. The sensing layer is vulnerable to sensors and devices which are targets of outcoming threats, including device tampering, spoofing attacks, signal attacks, malicious data, etc. At the middleware level, the secure transmission of sensed data and its storage are the primary concerns. This layer deals with confidentiality, integrity, and availability issues. The security requirements at the application layer are determined directly by the application being executed. Figure 2 presents a classification of possible security issues and their solutions for each of the fog architecture layers listed above [10].

The authors of [96] state that the most promising research directions for security solutions in fog computing are cryptographic techniques and machine-learning for intrusion detection. Cryptographic processing includes encryption, decryption, key and hash generation, and verification of hashes used to guarantee data privacy.

As an example of this technique, the Configurable Reliable Distributed Data Storage System [19] was designed to secure data flows in whole fog. Such a system uses the AR-RRNS (Approximation of the Rank – Redundant Residue Number System) method to encrypt and decrypt data using error correction codes and secret sharing schemes. Machine-learning techniques are proposed to analyze data flow and node states to detect outside intrusion. To implement such a traffic analysis, the fog orchestrator can act as a tool to detect an intrusion or data corruption [29].

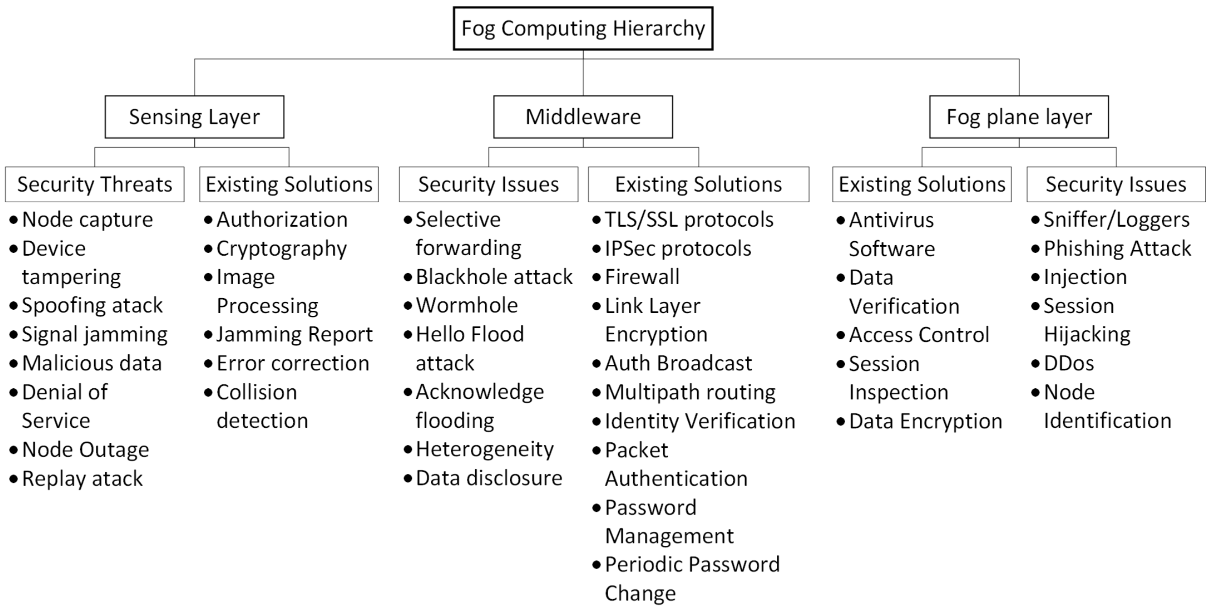

Figure 2. The security threats and solutions classifications in fog computing. DDoS: distributed DoS; TLS: transport layer security; SSL: secure sockets layer; IPsec: Internet Protocol security [10]

3.4. Fog Computing and Scalability

Scalability is another essential feature for fog computing systems to adapt workload, system cost, performance, and business needs. Based on fog computing hierarchical properties, we can highlight the following key elements of fog architecture that can be scaled [85]:

- *Virtual Nodes:* through software reconfiguration, specifying if several virtual nodes can be placed on one physical device;
- *Physical Nodes:* vertical scalability trough hardware upgrade;
- *Networks:* horizontal scaling of fog nodes and adapting to environmental changes and dynamic workloads.

Adding new fog nodes to the fog network affects all three main aspects of scalability discussed above (the question concerning fog storage resources won't be reviewed in this paper). However, this task commonly requires manual workload from network administrators, while it is hard to effectively identify the location or cluster of the new fog node. In [85] the fog model that helps to overcome this difficulty was proposed. It automates the introduction of new fog nodes into an existing network based on the current network conditions and services required by customers. Concretely, the newly added fog node can detect its geographical location (e.g., using its network scan capability or via its GPS module) and identify the most suitable cluster to connect with it.

Kubernetes platform is now the de-facto standard for service management in centralized distributed systems such as clouds. In this regard, its application to the management of fog infrastructures is of definite scientific interest. Kubernetes has a native mechanism for auto-scaling that considers only CPU usage. Users can specify the maximal number of application instances, but the actual number of application instances activated is under the control of

Kubernetes. Authors of [101] developed a modification of the scheduling algorithm based on the Kubernetes platform to manage resource autoscaling tasks in fog computing systems. A fog computing platform has been designed as a collection of physically distributed containers that are orchestrated by Kubernetes and AS Broker – a service running in a Pod on Master. It communicates through APIServer with the Controller and Scheduler of Kubernetes to obtain a list of nodes where application instances are currently running. It then collects node information from all nodes. If the number of application instances should be adjusted, it sends a request through APIServer for Pod number adjustment.

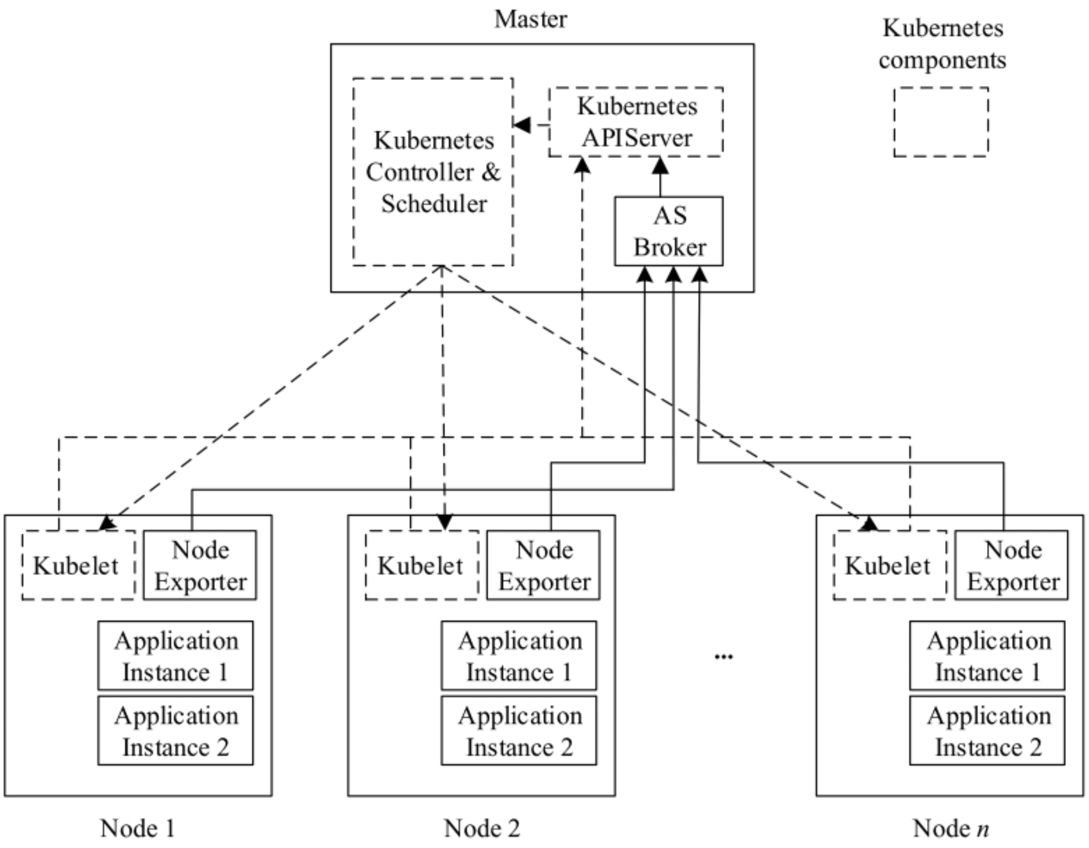

Figure 3. Proposed architecture of fog network based on Kubernetes [101]

The scalability experiment in [101] included four independent fog nodes. A stress program was used to generate CPU and memory load to emulate the processing of requests. Every request took a 15-second execution time and a 50 MB memory amount. Figure 4 shows tested application response time with and without AS Broker. Though response time dynamically changes, the result with AS Broker is better than that without almost at every time point. This result demonstrates the effectiveness of the proposed scheme.

The authors of [28] also investigate the possibilities of automatic scaling of computing resources of fog platforms developing their own Voilà platform. The objective of their work was to dynamically scale and place an applications replicas in a cluster of geo-distributed fog nodes to predominantly minimize the number of slow requests while maintaining efficient resource utilization. As a critical parameter determining the quality of service, the authors use the average response time parameter whether the latency and the processing capacity requirements are still

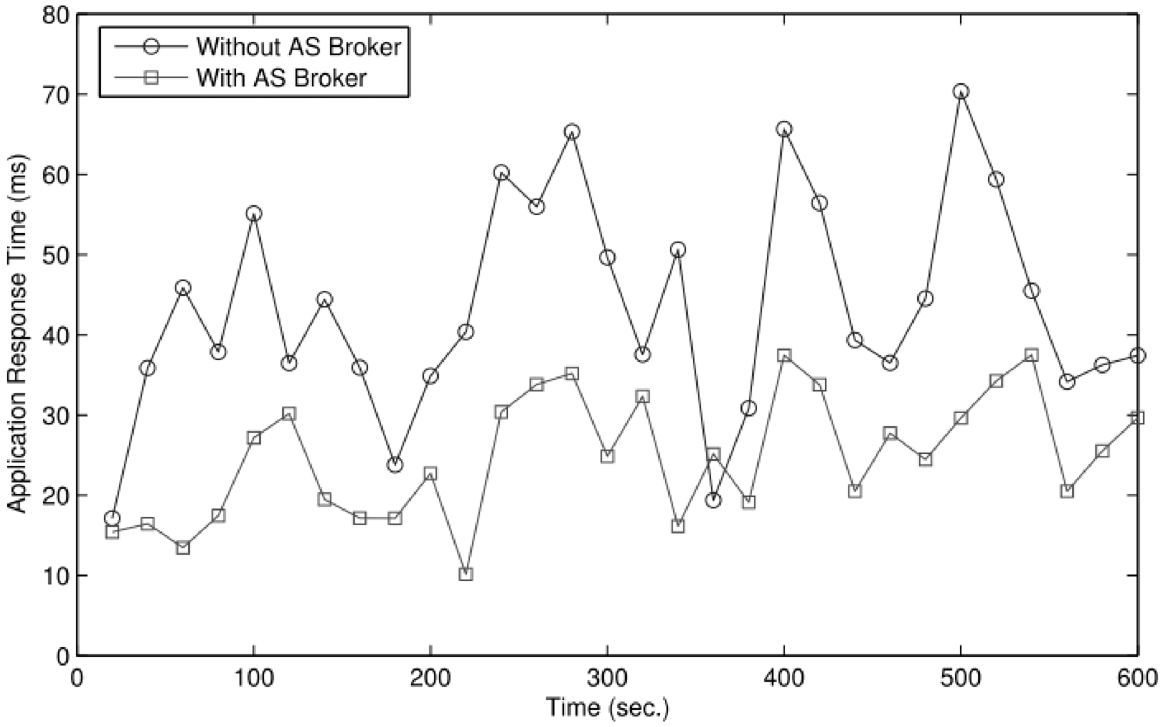

Figure 4. Application response time with and without AS Broker [101]

met. The following methods have been used to improve the quality of service: transferring service from one node to another, load redirection to the nearest lightly loaded servers, or scaling, by creating new replicas of computing services. The experimental setup consisted of 22 Raspberry Pi (RPi) model 3B+ single-board computers acting as fog computing servers. The RPis were organized with one master node and 21 worker nodes capable of hosting replicas. Node deployment was controlled with Kubernetes including node status, average number of placement,etc. as part of its normal operations. Replica placement quality evaluation was presented in [28] as main part of their experiment evaluations. Figure 5 shows the average number of placements that could be studied per second for various system sizes with the average number of placements that had to be evaluated to repair a latency or capacity violation. When the cluster size increased, the time needed to study any single placement also increased. However, even for a large system with 500 nodes, author's system evaluated approximately 100 placements per second.

The idea of using fog computing as a computing swarm and architecture of organizing fog nodes and application was described in [16]. In the proposed architecture, the fog consists of fog nodes (FNs), fog node controllers (FNCs), applications (Apps) and application controllers (ACs). FNCs control when FNs can be attached and detached. With the help of ACs FNCs can scale up and down the set of resources assigned to an App (e.g. by decreasing/increasing the cores, CPU time, and bandwidth assigned to such App) by simply changing the resources assigned to the corresponding Docker container. If the computational resources available in a FN are no more capable of satisfying the requirements of all Apps running on it, the FNC of the overloaded FN will interact with the other FNCs in the system to decide which Apps can be migrated and on which FNs.

Attempts to use fog computing with low-power devices to solve resource-intensive computational problems have been made since the beginning of the concept of fog computing. Authors

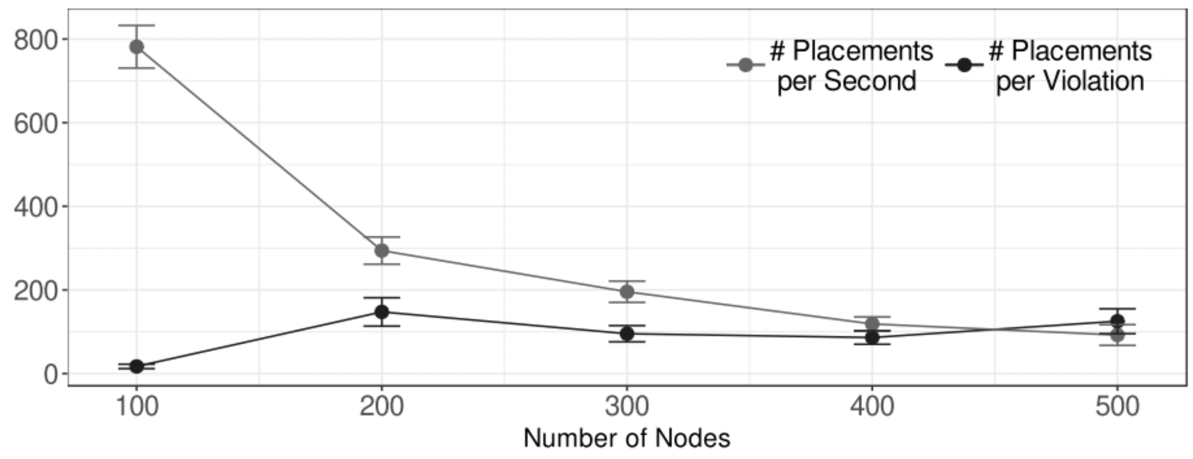

Figure 5. Evalutation of average number of placement that can be studied per second [28]

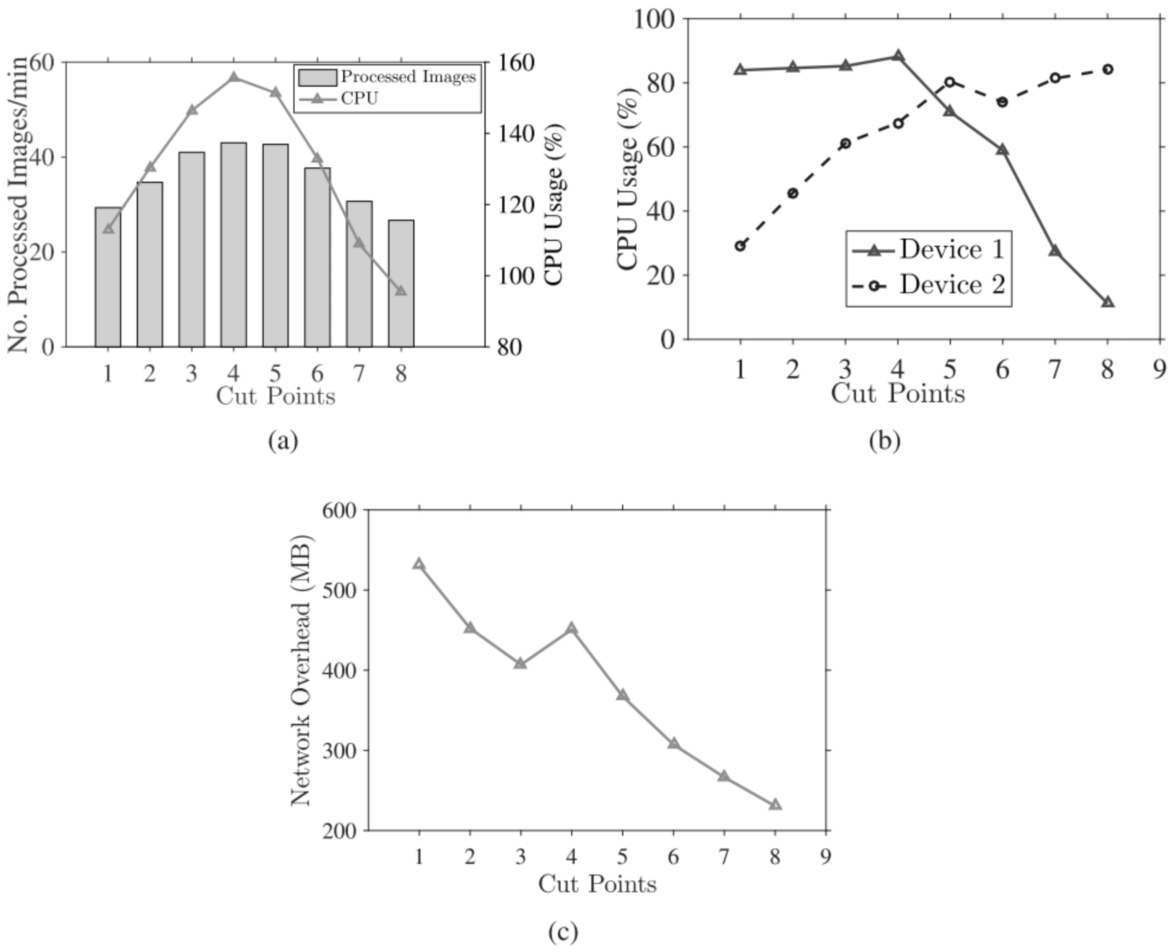

Figure 6. Different service quality caused by different cutting points: (a) the number of processed images, (b) the CPU and RAM usages, and (c) the network overhead [86]

of [86] use the resources of low-power Raspberry Pi-based nodes for Machine Learning Data Analytics using Deep Learning. They took the SCALE (Safe Community and Alerting Network) TensorFlow Application that uses various sensors (movement detection, gas, temperature, heartbeat, etc.) to build a security system as an example. The authors enhanced this project to support two crucial sensors: camera and microphone. They collected sensor data along the path where the person with the camera passed. Figure 6a shows the number of processed images per minute when running enhanced application on two devices with eight different cutting points. The application implemented a 9-layers network, so cuts are made between layers. When the cutting point went from 1 to 8, more complicated operators were put on the first device. The first device processed images before the second device. As shown in Fig. 6a, cutting points 4 and 5 resulted in the best performance. It is explained by Fig. 6b, which shows that cutting an application into smaller operators with similar complexity results in the best performance. Moreover, Fig. 6c reports the network overhead caused by distributed analytics. It shows that if more loads were put on the first device, it resulted in lower network overhead. Hence, when network resources were the bottleneck, equally-loaded splitting decisions were not preferred.

4. Overview of Fog Computing Platforms

While reviewing the existing fog computing deployment platforms, we would consider the commercial ones as well as open-source platforms. The complexity of the analysis of commercial platforms is the lack of information about their architecture and the technical solutions used, which constitute a trade secret. However, the analysis of commercial solutions has shown that among commercial fog platforms, there are platforms with the full support of fog computing (computing, analytics, and organization of the transport layer of the fog network) and platforms that provide only the transport layer of the fog network and do not provide management of computing nodes and fog computing itself. Platforms that provide only the transport layer of fog computing will not be considered in this paper.

The following key characteristics of private and public commercial fog platforms can be highlighted (see Tab. 3 and Tab. 4).

- *Supported hardware platforms* – the platform can work with any device that supports virtualization or containerization, or only with a limited list of devices – through drivers or branded devices. Smartly Fog, ThingWorx, and Cisco IOx only work with their proprietary hardware.
- *Basic development technology* – which executable environment is used to create, deploy and run fog applications.
- *Open communication protocols and SDK* – is there any restriction on the applications that can be used in the fog: whether it is necessary to port applications, or in principle can be executed only applications written using special supplied SDK, as in the case of ThingWorx, whose fog applications should be written using a proprietary SDK to run in the fog.
- *Deployment technology* – which of the technologies is used to deploy fog nodes, if known.
- Integration options – is it possible to integrate with other platforms, such as enterprise solutions or public clouds?
- *Connecting of external data sources* – the platform's ability to connect to third-party databases and data warehouses physically located outside the central cloud for data storage and processing.
- *Availability of additional services (Machine Learning, Analytics, etc.)* – the ability to connect and use additional services, which provide additional functionality for analysis and work with data in the fog.

- *Edge support* – the ability to connect and use edge devices and edge computing, and further collect and process information from them.

4.1. Private Fog Platforms

Private fog platforms provide private fog solutions based on computing infrastructure deployed directly on the customer's resources.

Table 3. Overview of private fog platforms

Feature	**ClearBlade**	**Smartiply Fog**	**LoopEdge**	**ThingWorx**	**Nebbiolo Technologies**	**Cisco IOx**
Supported hardware platforms	Universal	Own equipment	Universal	Own equipment	Universal	Own equipment
Basic development technology	JavaScript	No data	Universal (Docker)	Java VM	Universal (Docker)	Docker, Linux, IOx
Open communication protocols and SDK	+	+	+		+	+
Deployment technology	Linux KVM	No data	Docker	No data	Docker	Linux KVM
Integration opportunities	Oracle, SAP, Microsoft, Salesforce			Microsoft, Azure IoT, Hub		Microsoft, Azure IoT, Hub
Connecting external data sources	+		+	+	+	+
Availability of additional services	No data.	+		+	+	+
Edge Support	+	+	+	+	+	+

The Cisco IOx platform was presented by Cisco in 2014 [12] as a network infrastructure development due to the expected growth of IoT. The platform's focus is to reduce the labor costs of porting applications to the fog nodes, achieved through containerization technologies and based on its operating system based on the Linux OS.

The Cisco IOx is an application environment that combines Cisco IOS (a mini operating system of Cisco hardware) and Linux. Open-source Linux utilities are used to develop applications. It uses a single protocol for the interaction of fog applications throughout the network, organized using Cisco IoT technologies. Both Cisco and its partners supply IOx infrastructure fog applications. A variety of general-purpose programming languages is supported to develop Cisco IOx applications.

The Docker is used for deploying applications. Various types of applications are supported, including Docker containers and virtual machines (if network equipment has such a capability). It is also possible to use your IOx executable environment to write applications in high-level programming languages (such as Python).

The Nebbiolo Technologies platform is aimed at the corporate industrial market, supporting the Industry 4.0 concept [67]. Nebbiolo Technologies closely cooperates with Toshiba Digital Solutions [66] in supplying complete computing solutions for the industrial and IoT sectors.

The platform consists of fogNode hardware, fogOS software stack, and fogSM system administrator, deployed in the cloud or locally [45]. Fog System Manager (fogSM) provides a cloud-based, centralized management platform that allows you to deploy and configure devices on the periphery.

The platform's key feature is fogOS [45] – a software stack that provides communication, data management and application deployment at the fog level. Based on a hypervisor, fogOS provides a set of functions in a virtualized form. It supports a wide range of device connectivity standards and allows applications to be hosted and managed in real-time.

The ClearBlade Platform is a technology stack that provides fast development and deployment of enterprise IoT solutions, from edge devices to cloud services. It includes software components installed on the entire IoT device stack and the ability to connect third-party systems through the provided API for integration with devices, internal business applications, and cloud services. The ClearBlade Platform provides a centralized console for managing IoT applications, with the ability to deploy locally or in the cloud. Platform management functions are delegated to the edge nodes (or on the end devices themselves or their gateways) using ClearBlade Edge fog and edge computing [48].

The platform supports a serverless computing approach to the development of services based on the JavaScript language, which can be configured to implement machine learning and data analysis methods. The platform provides mechanisms for exporting data and analytics collected by the system to widely used business systems, applications, and databases through integration with corporate platform solutions from Oracle, SAP, Microsoft, and Salesforce. ClearBlade also provides in-house dashboards, business applications, and database management systems for integrated monitoring and management of the IoT ecosystem.

ClearBlade uses the OAuth model for access control, where each user and device receives a token that must be authorized to gain access to the system or its node. The data is encrypted on the devices themselves as well as on network transmissions. Transmitted data is encrypted using OpenSSL libraries with TLS encryption.

The Smartiply Fog platform is a cloud computing platform that focuses on optimizing resources and keeping your devices running even without connecting to the cloud. The platform provides greater reliability for online environments by optimizing resources and computing based on proprietary hardware [82]. The platform enables point-to-point interaction between devices. In this way, the node system can continue to operate autonomously to receive, analyze and store data, up to restoring communication with the external network [83].

The LoopEdge platform from Litmus Automation [4, 6] allows to connect different devices in a single system, collect and analyze data from them. Litmus Automation also provides a Loop platform allowing to manage the life cycle of any IoT device and export real-time data to internal analytical and business applications. This platform is widely distributed among well-known engineering concerns: Nissan, Renault, Mitsubishi Corporation Techno.

The platform developers emphasize that it can work with virtually any device, industrial and domestic consumers. For example, the platform supports the connection of devices based on Arduino and Raspberry Pi. Even if some device is not supported, connecting it to the platform is relatively easy due to the executable packages installed on the device itself, which can be expanded and created from scratch for a particular device.

The PTC ThingWorx platform [7] is an IoT platform that offers the connection possibility to more than 150 types of devices. However, since devices are connected through drivers that require installation, this platform is not universal and has limitations on the devices used.

Applications for the platform should be written using the supplied SDKs. Further data analysis and business process management also go through the tools provided by the platform itself. The platform has an extensive developer section with instructions, tutorials, and assistance from specialists from the company itself to install, configure, and expand the platform. Also "out of the box" is the possibility of connecting to Microsoft Azure IoT Hub.

4.2. Public Fog Platforms

Table 4. Overview of public fog platforms

Feature	AWS Greengrass	Azure IoT	Google	Yandex	Mail.ru
Supported hardware platforms	Universal	Universal	Universal	Universal	Universal
Basic development technology	Universal (Docker)	Universal (Docker)	Universal	Universal	Universal
Open communication protocols and SDK	+	+	+	+	+
Deployment technology	Docker	Docker	Docker	Docker	Docker
Integration capability	Amazon Elastic Compute 2	Azure, via an API	Services of Google and partners, through API	Universally via API	Universally via API
Connecting external data sources			+	+	
Availability of additional services (Machine Learning, Analytics, etc.).	+	+	+	+	+
Support Edge	+	+	+	+	+

Today, public fog platforms are the solutions of major players in the fog computing market, focused on solving data processing tasks from IoT systems linked to the capabilities of the corresponding cloud platform. The key characteristics of the considered public fog platform are given in Tab. 4.

The Azure IoT platform provides a platform for fog and edge computing based on Microsoft's technology stack. It consists of several extensive subsystems such as IoT Central and IoT Edge, which base their work on Microsoft Azure cloud technology. Connection of devices from Microsoft partners is possible without using drivers or software code due to IoT Plug and Play technology. This approach is possible for devices running any OS, including Linux, Android, Azure Sphere OS, Windows IoT, RTOS, and others.

Creation, installation, and management of fog applications are performed through the Azure IoT Hub portal. The IoT Hub is a cloud-based managed service that acts as a central message processor for bidirectional communication between an IoT application and the devices it manages. IoT Hub supports both device-to-cloud and cloud-to-device transfers. The IoT Hub supports multiple messaging templates, such as telemetry between devices and the cloud, downloading files from devices, and query-answer technology for managing devices from the cloud.

To deploy computing closer to the devices themselves or on the devices themselves, Azure IoT Edge system is used, allowing to deploy applications with their business logic, or already available in the directory ready-made applications on end devices using containerization technology.

The Amazon AWS IoT Greengrass platform allows to extend the capabilities of AWS (Amazon Web Services) to one's peripherals, enabling them to work locally with one's data while using the cloud to manage, analyze and securely store one's data. AWS IoT Greengrass allows connected devices to perform AWS Lambda functions, run Docker containers, generate forecasts based on machine learning models, synchronize these devices and interact securely with other devices even without an Internet connection.

AWS IoT Greengrass allows to create IoT solutions that connect different types of devices to the cloud and each other. AWS IoT Greengrass Core can be used on Linux devices (including Ubuntu and Raspbian distributions) that support Arm or x86 architectures. The AWS IoT Greengrass Core service provides local AWS Lambda code execution, messaging, data management, and security. Devices with AWS IoT Greengrass Core serve as portals of the service and interact with other devices that run FreeRTOS (Real-time operating system for microcontrollers) or installed SDK package AWS IoT for devices. The size of such devices can be very different: from small devices based on microcontrollers to large household appliances. When a device with AWS IoT Greengrass Core loses contact with the cloud, devices in the AWS IoT Greengrass group can continue to communicate with each other over a local network.

Google, Yandex, and Mail.ru platforms provide their cloud and fog solutions for data collection, storage, processing, analysis, and visualization. Collected data from devices is integrated into the public cloud system for deeper processing and analysis (including machine learning and artificial intelligence) due to the high computing power of the cloud. These platforms support multiple protocols for connectivity and communication through the provided API. There are many ready-to-use services available for installation in the platform directory itself, which can be connected to your cloud solution by combining them.

4.3. Open Source Fog Platforms

During the analysis of existing solutions, we reviewed existing open-source fog platforms. In contrast to commercial solutions, for open-source platforms, there are complete descriptions of architectures, requirements to computing resources, as well as technologies used, both on hardware and software levels (see Tab. 5). Open Source approach often implies that technologies used to develop, maintain and deploy systems are free and available to contributors.

The FogFrame2.0 is an open-source fog platform [3] aimed at deployment on single-board computers (Raspberry Pi). Authors designed architecture and implemented a representative framework to resolve the following challenges [79]:

- enable the coordinated cooperation among computational, storage, and networking resources in the fog [80, 81];
- implement heuristic algorithms for service placement in the fog, namely a first-fit algorithm and a genetic algorithm;
- introduce mechanisms for adapting to dynamic changes in the fog landscape and for recovering from overloads and failures.

To evaluate the behavior of FogFrame, authors apply different arrival patterns of application requests, i.e., constant, pyramid, and random walk, and observe service placement. The platform dynamically reacts to events at runtime, i.e. when new devices appear or are disabled when devices experience failures or overloads, necessary node redeployments are performed.

The FogFlow platform is an open-source fog platform [2]. The developers' main task was to provide a flexible and straightforward way of development, deployment, and orchestration of fog services [15]. The uniqueness of their approach is as follows:

- standard-based programming model for fog computing with declarative hints;

- scalable context management: to overcome the limitations of centralized context management, FogFlow introduces a distributed context management approach.

The data structure of all data flows is described based on the same standardized data model called the NGSI. Therefore, FogFlow can learn which type of data is created at which edge node. It then triggers and launches dynamic data processing flows for each edge node based on the availability of registered context metadata, which gives service developers two advantages:

- fast and easy development of fog computing applications, because the proposed hints hide configuration and deployment complexity tasks from service developers;
- good openness and interoperability for information sharing and data source integration with the use of the NGSI – a standardized open data model and API and it has been widely adopted by more than 30 cities worldwide.

FogFlow is one of the components of FIWARE open infrastructure [32], which provides the development and implementation of various smart solutions [18, 27, 30]. This infrastructure is one of the modern cloud frameworks along with Amazon Web Services [39]. A wide library of ready-made solutions from the developer community and detailed implementation instructions are available for implementation and use of FogFlow [1].

The FogBus platform (supported by Melbourn Clouds Lab) integrates various hardware tools through software components that provide structured interaction and platform-independent application execution [88]. FogBus uses blockchain to ensure data integrity when transmitting sensitive data. The platform-independent architecture of application execution and interaction between nodes allows to overcome heterogeneity in the integrated environment.

FogBus supports implementing various resource management and scheduling policies to run IoT applications compiled using parallel programming models such as SPMD (single program, multiple data).

To evaluate the performance of the FogBus platform, a prototype application system is used to analyze the Sleep Apnea data. This example illustrates how an application (in the healthcare sector) built using the SPMD model can be implemented using different FogBus settings to process IoT data in an integrated computing environment.

This framework makes it easy to deploy IoT applications, monitor and manage resources. FogBus system services are developed in cross-platform programming languages (PHP and Java). Thay are used with the Extensible Application Layer Protocol (HTTP), which helps FogBus overcome heterogeneity in the communication level of the OS and P2P of different nodes of fog. Besides, the FogBus platform functions as a “Platform as a Service” (PaaS) model for the Fog Cloud integrated environment, which not only helps application developers create different types of IoT applications but also supports users to configure services, and service providers to manage resources according to system conditions without maintaining the infrastructure.

Table 5. Overview of Open Source Fog Platforms

	Goal	**Deployment**
FogFrame2.0	Check the conceptual model	
FogFlow	Simpler and more flexible orchestration of services	+
FogBus	Overcome heterogeneity at the communication level between OS and P2P of different nodes of the fog	

4.4. Classification Methods for Fog Platforms

To form a unified approach to the classification of fog platforms, we have considered the key fog platforms and their key characteristics. For example, AWS Greengrass can work without access to the public cloud[4], but it is possible only to store local data in this mode of operation. Central device management, as well as centralized data collection and processing, becomes impossible. The Entire platform operation requires access to AWS IoT Core, which acts as a central service for the management and organization of fog and a public cloud.

Azure IoT can also operate on private networks[5], but only if there is a gateway within the private network that must connect to the central management and data collection node, and that node is also a public cloud. What distinguishes IoT from a public cloud is that it has a single point of access to the external network, rather than many different gateways that communicate with the public cloud.

Other public fog platforms have the same limitations as private and open-source fog platforms, the central control node of which can be deployed on any server in the local network or not at all (in this case, management and orchestration tasks are separated between intermediate fog nodes, as is done, for example, in FogFlow2.0).

Therefore, all fog platforms can be classified according to the openness or closedness of the deployment of the hub, a service that is responsible for connecting, monitoring and managing devices connected in the fog. In one form or another, almost all commercial fog platforms has the hub: LoopEdge and Azure IoT call this service – the Hub. ClearBlade and FogHorn platforms have a service with the same functionality, but it is called Device Manager. At AWS Greengrass, this service is called AWS IoT Core.

Another criterion for classification may be the requirements for the underlying hardware on which fog platform services can be deployed. Some of these platforms are tightly bound to a limited list of supported hardware devices. On the other hand, other platforms allow their services to be deployed on any end-user hardware as long as it meets the necessary minimum requirements for platform deployment. We define this characteristic as an indicator of the classification of fog platforms according to the openness of the hardware infrastructure.

The same principle is observed when comparing platforms based on openness or closed software infrastructure: the platform can support open protocols of data exchange between nodes of fog or fog programs are supplied exclusively by the developers of the platform itself and licensed partners.

Thus, any fog platform can be classified according to the principle of openness or closeness of its components (see Fig. 2). It should also be noted that platforms with a public hub are more likely to be open to their hardware and software infrastructures.

In addition to the openness or closeness of their components, some platforms have focused on the availability of the various features or services provided by the platform. The Azure IoT Hub, which is an integral part of the Azure IoT platform, explicitly calls itself **PaaS (Platform as a Service)**, providing ready-made solutions for the user's required tasks. It should be noted that none of the public fog platforms positions their platforms as pure fog. They provide fog computing as a certain basic functionality, which is the basis for other provided platform functions and services.

[4]`https://aws.amazon.com/ru/greengrass/faqs/Local_Resource_Access`

[5]`https://azure.microsoft.com/en-us/blog/introducing-iot-hub-device-streams-in-public-preview/`

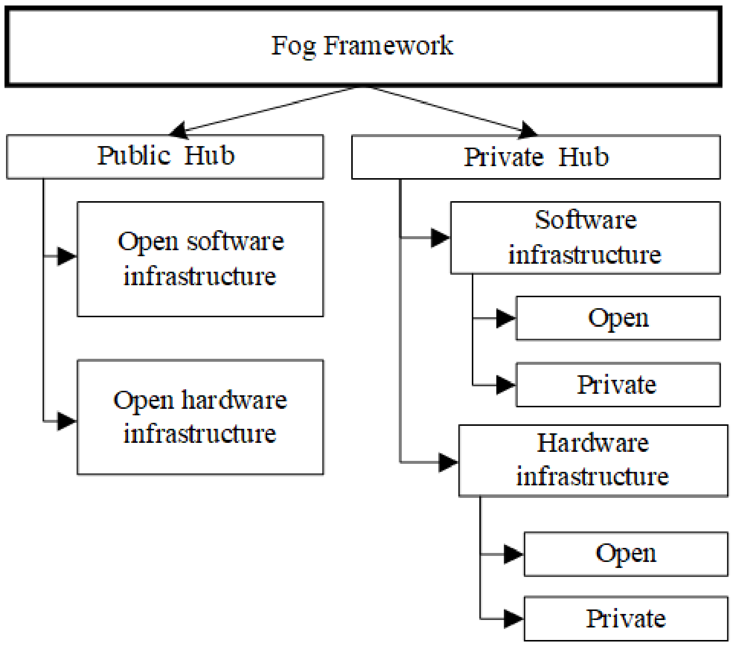

Figure 7. Classification of fog platforms according to the principle of openness or closeness of its components

Thus, the platforms themselves position some functionality as basic, which should be in any fog platform, and the user is interested not only in simple deployment and basic management of fog nodes but also in solving their specific tasks: Industry 4.0, Medicine, Smart City, etc. Platforms should provide ready-made solutions for each of the user's tasks as much as possible.

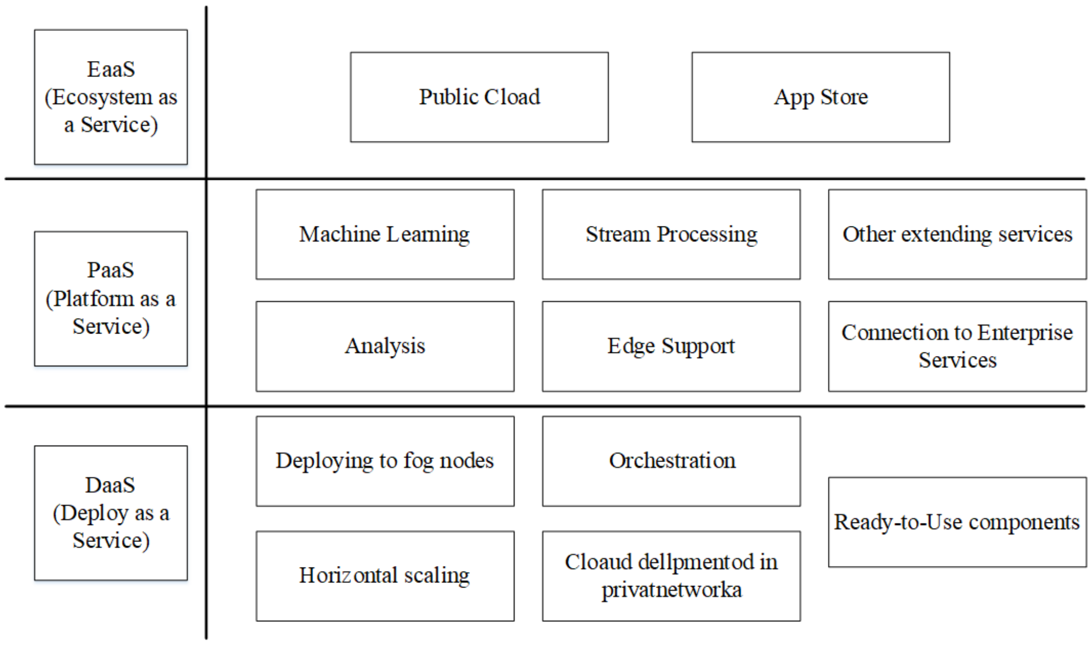

Figure 8. Classification of fog platforms based on provided platform functionality

Among other things, some platforms have allowed users to share their ready-made solutions created within the platform with the help of "stores" – resources where the user can publish his readymade fog application. This has led to the emergence of entire fog ecosystems – **EaaS**

Table 6. Reviewed platform classification

	Classification "as a service"	Hub	Software Infrastructure	Hardware Infrastructure
FogFrame2.0	DaaS	No Hub	Public	Public
FogFlow	PaaS	Private	Public	Public
FogBus	PaaS	Private	Public	Public
AWS Greengrass	EaaS	Public	Public	Public
Azure IoT	EaaS	Public	Public	Public
Google	EaaS	Public	Public	Public
Yandex	EaaS	Public	Public	Public
Mail.ru	EaaS	Public	Public	Public
ClearBlade	PaaS	Private	Private	Public
Smartiply Fog	PaaS	Private	Private	Private
LoopEdge	PaaS	Private	Public	Public
ThingWorx	PaaS	Private	Private	Private
Nebbiolo	PaaS	Private	Public	Public
Cisco IOx	PaaS	Private	Public	Private

(Ecosystem as a Service), which allow users to create their fog solution from ready-made components available on the platform.

This description also includes Open Source solutions that provide only a basic level of functionality – **DaaS (Deploy as a Service)**: deployment of fog nodes on existing devices, orchestration, etc. On the other hand, FogFlow has wider functionality and even its ecosystem, which includes ready-to-install components from both platform developers and the community.

Classification "as a service" can be used as a classification method based on the provided platform functionality (see Fig. 8).

This classification then can be applied to reviewed in this paper platforms (Tab. 6). Assuming that term "public" to hub means if the hub can be publicly accessed or "private" if there's only an option of deploying your own hub without ability to share your solution with others. "Private" software means that platform uses only limited list of applications and "public" if any software can be installed without publishing your application to some centralized hub. "Private" hardware if fog nodes can be deployed only on specific hardware and "public" hardware infrastructure can use a wide range of devices that meet requirements.

5. Fog Computing Challenges

In this section, we discuss several future research directions that are considered to be most promising for future research in other works and in this research likewise.

Artificial Intelligence Application Management is currently receiving considerable attention because of its ability to solve complex problems. The data needed to build an AI system is quickly accumulated in a fog [57]. Artificial Intelligence application management can help predict future resource requirements, context variations, and node failures more accurately and manage applications accordingly.

Fog nodes are limited in resources. Adding more fog nodes to the fog may reduce this limitation. However, it increases the cost of deployment, complexness of node communication, and power consumption at the network edge [8]. In this case, it may be helpful to **dynamically consolidate and scale the fog nodes** according to computational and storage needs. Fog computing is developed to execute various complex IoT applications from different domains, including smart healthcare, city, agriculture, and industry [64]. These IoT applications have specific requirements and the need for specialized support. **Application-specific management strategies** can help deal with them in Fog.

Task and data processing is decentralized in the Fog. The task may begin on one node and going through several others end on the last one. When an emergency happens in the fog, the developer and fog designer need access to log information to locate the problem in minimal time. Thus total logging helps with this task, but then the problem appears to maintain a data lake supporting storage and analysis of such data. Thats the question of **logging and monitoring of highly-distributed fog applications**.

On the other hand, there is also the issue of **task sharing and re-usability**. Applications can share a particular task to optimize the computational load on fog nodes [90]. Besides, the task executables of recently terminated applications can also be reused for other applications. To perform such operations, shared caching techniques and policies are required to be developed in the context of fog computing.

The above opens another question. If the fog node faces a software or hardware problem and shuts down other nodes wont have any information on the checkpoint the node was in. But most applications are state-dependent and stateful. So there is a challenge to organize **state management and sharing** between nodes to support the continuous and flowless work of the fog.

Most Fog applications do not consider security as part of a system but rather focus on functionality, which results in many fog platforms being vulnerable [54]. That leads to sensitive data leakage, user loss of privacy, and other **security issues** that are very significant in most IoT domains. Future work could lead towards the development of knowledge-based supplementary references, which can provide decision support for developers in designing a secure and performance efficient fog infrastructure. Such decision support would require a large systematic knowledge acquisition of best practices, known security threats and their solutions, which can be formalized as either a statistical-based system or rules, policies and facts.

Conclusion

The increase of transferred data volumes and the increased load on the cloud for client services became a prerequisite for the concept of fog computing. In this paper, the concept of fog computing, its definition and key characteristics were considered. Also, there were considered, classified and generalized some fog platforms, which are subjects of research or already used by business and private clients. In the end, the general architectural characteristics inherent in all the platforms reviewed were described.

Fog computing is a more flexible and efficient type of computation compared to cloud computing due to the solution of tasks requiring high bandwidth of the computing network, the ability to work with geographically dispersed data sources, ultra-low latency and providing local data processing. In this review paper, we not only gave an extended point of view over the fog computing paradigm but also analyzed the growing diverse number of open source and enterprise solutions for deploying fog platforms. On the basis of this review, we proposed a classification of fog solutions by their cloud layer, hardware and software publicity level and by a provided service and functionality they grant.

Acknowledgements

The research was carried out with the financial support of the RFBR and the Chelyabinsk Region, project number 20-47-740005 (chapters 1–3) and the Ministry of Science and Higher Education of the Russian Federation (state assignment FENU-2020-0022) (chapters 4, 5).

References

1. FogFlow - FogFlow v2.0 documentation. https://fogflow.readthedocs.io/en/latest/, accessed: 2020-02-27
2. Smartfog/fogflow: FogFlow is a standard-based IoT fog computing framework that supports serverless computing and edge computing with advanced programming models. https://github.com/smartfog/fogflow, accessed: 2020-02-27
3. Softls/fogframe-2.0: Fogframe framework (with extensions). https://github.com/softls/FogFrame-2.0, accessed: 2020-02-27
4. Litmus Automation Releases Next Generation of LoopEdge Industrial IoT Gateway Software. https://litmus.io/loopedge-update-release/ (2017), accessed: 2020-02-27
5. IDC's Global DataSphere forecast shows continued steady growth in the creation and consumption of data. https://www.idc.com/getdoc.jsp?containerId=prUS46286020 (2020), accessed: 2020-06-20
6. Litmus Automation - Platform Features. https://litmus.io/litmus-edge/features/ (2020), accessed: 2020-02-27
7. PTC Thingworx. https://www.ptc.com/ru/products/thingworx (2020), accessed: 2020-02-27
8. Afrin, M., Jin, J., Rahman, A., et al.: Multi-objective resource allocation for Edge Cloud based robotic workflow in smart factory. Future Generation Computer Systems 97, 119–130 (2019). https://doi.org/10.1016/j.future.2019.02.062
9. Al-Doghman, F., Chaczko, Z., Ajayan, A.R., Klempous, R.: A review on Fog Computing technology. In: 2016 IEEE International Conference on Systems, Man, and Cybernetics, SMC 2016 - Conference Proceedings. pp. 1525–1530. IEEE (2017). https://doi.org/10.1109/SMC.2016.7844455
10. Aljumah, A., Ahanger, T.A.: Fog computing and security issues: A review. In: 2018 7th International Conference on Computers Communications and Control, ICCCC. pp. 237–239. IEEE (2018). https://doi.org/10.1109/ICCCC.2018.8390464

11. Anawar, M.R., Wang, S., Azam Zia, M., et al.: Fog computing: An overview of big IoT data analytics. Wireless Communications and Mobile Computing 2018 (2018). https://doi.org/10.1155/2018/7157192

12. Antonio, S.: Cisco delivers vision of fog computing to accelerate value from billions of connected devices pp. 1–4 (2014)

13. Armbrust, M., Fox, A., Griffith, R., et al.: A view of cloud computing. Communications of the ACM 53(4), 50–58 (2010). https://doi.org/10.1145/1721654.1721672

14. Bellendorf, J., Mann, Z.Á.: Classification of optimization problems in fog computing. Future Generation Computer Systems 107, 158–176 (2020). https://doi.org/10.1016/j.future.2020.01.036

15. Bonomi, F., Milito, R., Zhu, J., Addepalli, S.: Fog computing and its role in the Internet of Things. In: MCC'12 - Proceedings of the 1st ACM Mobile Cloud Computing Workshop. pp. 13–15. ACM Press, Helsinki (2012). https://doi.org/10.1145/2342509.2342513

16. Brogi, A., Mencagli, G., Neri, D., et al.: Container-based support for autonomic data stream processing through the fog. In: Euro-Par 2017: Parallel Processing Workshops. Euro-Par 2017. Lecture Notes in Computer Science, vol. 10659, pp. 17–28. Springer Verlag (2018). https://doi.org/10.1007/978-3-319-75178-8_2

17. Brynjolfsson, E., Hofmann, P., Jordan, J.: Cloud computing and electricity: Beyond the utility model. Communications of the ACM 53(5), 32–34 (2010). https://doi.org/10.1145/1735223.1735234

18. Celesti, A., Fazio, M., Márquez, F.G., et al.: How to develop IoT cloud e-health systems based on FIWARE: A lesson learnt. Journal of Sensor and Actuator Networks 8(1) (2019). https://doi.org/10.3390/jsan8010007

19. Chervyakov, N., Babenko, M., Tchernykh, A., et al.: AR-RRNS: Configurable reliable distributed data storage systems for Internet of Things to ensure security. Future Generation Computer Systems 92, 1080–1092 (2019). https://doi.org/10.1016/j.future.2017.09.061

20. Chiang, M., Ha, S., Risso, F., et al.: Clarifying fog computing and networking: 10 questions and answers. IEEE Commun. Mag. 55(4), 18–20 (2017). https://doi.org/10.1109/MCOM.2017.7901470

21. Cisco Systems: Fog computing and the Internet of Things: Extend the cloud to where the things are. http://www.cisco.com/c/dam/en_us/solutions/trends/iot/docs/computingoverview.pdf (2016), accessed: 2020-02-27

22. Dantas, L., Cavalcante, E., Batista, T.: A Development Environment for FIWARE-based Internet of Things Applications. In: M4IoT 2019 - Proceedings of the 2019 Workshop on Middleware and Applications for the Internet of Things, Part of Middleware 2019 Conference. pp. 21–26. ACM, Davis CA (2019). https://doi.org/10.1145/3366610.3368100

23. Dar, B.K., Shah, M.A., Islam, S.U., et al.: Delay-aware accident detection and response system using fog computing. IEEE Access 7, 70975–70985 (2019). https://doi.org/10.1109/ACCESS.2019.2910862

24. De Brito, M.S., Hoque, S., Magedanz, T., et al.: A service orchestration architecture for Fog-enabled infrastructures. In: 2017 2nd International Conference on Fog and Mobile Edge Computing, FMEC 2017. pp. 127–132. IEEE, Valencia (2017). https://doi.org/10.1109/FMEC.2017.7946419

25. De Donno, M., Tange, K., Dragoni, N.: Foundations and Evolution of Modern Computing Paradigms: Cloud, IoT, Edge, and Fog. IEEE Access 7, 150936–150948 (2019). https://doi.org/10.1109/ACCESS.2019.2947652

26. Eugene, G.: Cloud computing models. Tech. Rep. January (2013). https://doi.org/10.1201/b11149

27. Evans, D.: The Internet of Things: How the next evolution of the internet is changing everything. CISCO white paper 1, 1–11 (2011)

28. Fahs, A.J., Pierre, G., Elmroth, E.: Voilà: Tail-latency-aware fog application replicas autoscaler. In: Proceedings - IEEE Computer Society's Annual International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems, MASCOTS. pp. 1–8. IEEE Computer Society (2020). https://doi.org/10.1109/MASCOTS50786.2020.9285953

29. Fakude, N.C., Tarwireyi, P., Adigun, M.O., Abu-Mahfouz, A.M.: Fog orchestrator as an enabler for security in fog computing: A review. In: Proceedings - 2019 International Multidisciplinary Information Technology and Engineering Conference, IMITEC 2019. pp. 1–6. IEEE (2019). https://doi.org/10.1109/IMITEC45504.2019.9015896

30. Fazio, M., Celesti, A., Marquez, F.G., et al.: Exploiting the FIWARE cloud platform to develop a remote patient monitoring system. In: Proceedings - IEEE Symposium on Computers and Communications. pp. 264–270. IEEE (2016). https://doi.org/10.1109/ISCC.2015.7405526

31. Feeney, G.J.: Utility computinga superior alternative? In: AFIPS Conference Proceedings - 1974 National Computer Conference, AFIPS 1974. pp. 1003–1004. ACM Press, Chicago (1974). https://doi.org/10.1145/1500175.1500370

32. FIWARE: What is FIWARE? (2015), https://www.fiware.org/about-us/

33. Foster, I.T., Kesselman, C.: The history of the grid. In: High Performance Computing: From Grids and Clouds to Exascale - Selected Papers from the High Performance Computing Workshop. Advances in Parallel Computing, vol. 20, pp. 3–30. IOS Press (2010). https://doi.org/10.3233/978-1-60750-803-8-3

34. Garcia, J., Simo, E., Masip-Bruin, X., et al.: Do we really need cloud? estimating the fog computing capacities in the city of Barcelona. In: Proceedings - 11th IEEE/ACM International Conference on Utility and Cloud Computing Companion, UCC Companion 2018. pp. 290–295. IEEE (2019). https://doi.org/10.1109/UCC-Companion.2018.00070

35. GE Digital: What is edge computing? (2018), https://www.ge.com/digital/blog/what-edge-computing

36. González, L.M.V., Rodero-Merino, L.: Finding your way in the fog: Towards a comprehensive definition of fog computing. Comput. Commun. Rev. 44(5), 27–32 (2014). https://doi.org/10.1145/2677046.2677052

37. Gu, L., Zeng, D., Guo, S., Barnawi, A., Xiang, Y.: Cost efficient resource management in fog computing supported medical cyber-physical system. IEEE Transactions on Emerging Topics in Computing 5(1), 108–119 (2017). https://doi.org/10.1109/TETC.2015.2508382

38. Guevara, J.C., Torres, R.d.S., da Fonseca, N.L.: On the classification of fog computing applications: A machine learning perspective. Journal of Network and Computer Applications 159 (2020). https://doi.org/10.1016/j.jnca.2020.102596

39. Guth, J., Breitenbucher, U., Falkenthal, M., et al.: Comparison of IoT platform architectures: A field study based on a reference architecture. In: 2016 Cloudification of the Internet of Things, CIoT 2016. pp. 1–6. IEEE (2017). https://doi.org/10.1109/CIOT.2016.7872918

40. Hagiu, A., Wright, J.: When data creates competitive advantage ... ... and when it doesn't. Harvard Business Review 98(1), 94–101 (2020)

41. Hannabuss, S.: The Big Switch: Rewiring the World, from Edison to Google. Library Review 58(2), 136–137 (2009). https://doi.org/10.1108/00242530910936989

42. Haouari, F., Faraj, R., Alja'Am, J.M.: Fog computing potentials, applications, and challenges. In: 2018 International Conference on Computer and Applications, ICCA 2018. pp. 399–406. IEEE, Beirut (2018). https://doi.org/10.1109/COMAPP.2018.8460182

43. Hashemi, S.M., Bardsiri, A.K.: Cloud computing vs. grid computing. ARPN Journal of Systems and Software 2(5), 188–194 (2012)

44. Hofmann, P., Woods, D.: Cloud computing: The limits of public clouds for business applications. IEEE Internet Computing 14(6), 90–93 (2010). https://doi.org/10.1109/MIC.2010.136

45. Hong, C.H., Varghese, B.: Resource management in fog/edge computing: A survey on architectures, infrastructure, and algorithms. ACM Computing Surveys 52(5), 1–37 (2019). https://doi.org/10.1145/3326066

46. Hong, H.J.: From cloud computing to fog computing: Unleash the power of edge and end devices. In: Proceedings of the International Conference on Cloud Computing Technology and Science, CloudCom. pp. 331–334. IEEE Computer Society, Hong Kong (2017). https://doi.org/10.1109/CloudCom.2017.53

47. Huang, C., Lu, R., Choo, K.K.R.: Vehicular fog computing: Architecture, use case, and security and forensic challenges. IEEE Communications Magazine 55(11), 105–111 (2017). https://doi.org/10.1109/MCOM.2017.1700322

48. Hughes, I., Immerman, D., Daly, P.: ClearBlade demonstrates scalability and edge analytics with IoT platform. Tech. rep. (2017)

49. IBM: What is fog computing? https://www.ibm.com/blogs/cloud-computing/2014/08/fog-computing/ (2016), accessed: 2020-02-27

50. Industrial Internet Consortium: The Industrial Internet Consortium and Openfog Consortium Join Forces. https://www.iiconsortium.org/press-room/01-31-19.htm (2019), accessed: 2020-02-27

51. Iorga, M., Feldman, L., Barton, R., et al.: Fog computing conceptual model. Tech. rep., Gaithersburg (2018). https://doi.org/10.6028/NIST.SP.500-325

52. Jiang, Y., Huang, Z., Tsang, D.H.K.: Challenges and solutions in fog computing orchestration. IEEE Network 32(3), 122–129 (2018). https://doi.org/10.1109/MNET.2017.1700271

53. Kakakhel, S.R.U., Mukkala, L., Westerlund, T., et al.: Virtualization at the network edge: A technology perspective. In: 2018 3rd International Conference on Fog and Mobile Edge Computing, FMEC 2018. pp. 87–92. IEEE, Barcelona (2018). https://doi.org/10.1109/FMEC.2018.8364049

54. Khan, S., Parkinson, S., Qin, Y.: Fog computing security: a review of current applications and security solutions. Journal of Cloud Computing 6(1) (2017). https://doi.org/10.1186/s13677-017-0090-3

55. Kumar, R., Charu, S.: An importance of using virtualization technology in cloud computing. Global Journal of Computers & Technology 1(2), 56–60 (2015)

56. Li, C., Xue, Y., Wang, J., et al.: Edge-oriented computing paradigms: A survey on architecture design and system management. ACM Computing Surveys 51(2), 39:1–39:34 (2018). https://doi.org/10.1145/3154815

57. Li, H., Ota, K., Dong, M.: Deep reinforcement scheduling for mobile crowdsensing in fog computing. ACM Transactions on Internet Technology 19(2), 1–18 (2019). https://doi.org/10.1145/3234463

58. Liu, L., Wang, Y., Yang, Y., Tian, Z.: Utility-based computing model for grid. In: 2005 International Conference on Semantics, Knowledge and Grid, SKG 2005. p. 109. IEEE Computer Society (2005). https://doi.org/10.1109/SKG.2005.140

59. Liu, Y., Fieldsend, J.E., Min, G.: A framework of fog computing: Architecture, challenges, and optimization. IEEE Access 5, 25445–25454 (2017). https://doi.org/10.1109/ACCESS.2017.2766923

60. Madsen, H., Albeanu, G., Burtschy, B., Popentiu-Vladicescu, F.: Reliability in the utility computing era: Towards reliable fog computing. In: International Conference on Systems, Signals, and Image Processing. pp. 43–46. IEEE Computer Society, Rio de Janeiro (2013). https://doi.org/10.1109/IWSSIP.2013.6623445

61. Mahmood, Z., Ramachandran, M.: Fog computing: Concepts, principles and related paradigms. In: Fog Computing: Concepts, Frameworks and Technologies, pp. 3–21. Springer (2018). https://doi.org/10.1007/978-3-319-94890-4_1

62. Mahmoudi, C., Mourlin, F., Battou, A.: Formal definition of edge computing: An emphasis on mobile cloud and IoT composition. In: 2018 3rd International Conference on Fog and Mobile Edge Computing, FMEC 2018. pp. 34–42. IEEE, Barcelona (2018). https://doi.org/10.1109/FMEC.2018.8364042

63. Mell, P., Grance, T.: The NIST definition of cloud computing: Recommendations of the National Institute of Standards and Technology. In: Public Cloud Computing: Security and Privacy Guidelines, pp. 97–101 (2012)

64. Mohamed, N., Al-Jaroodi, J., Jawhar, I.: Towards fault tolerant fog computing for IoT-based smart city applications. In: 2019 IEEE 9th Annual Computing and Communication Workshop and Conference, CCWC 2019. pp. 752–757. IEEE (2019). https://doi.org/10.1109/CCWC.2019.8666447

65. Naha, R.K., Garg, S., Georgakopoulos, D., et al.: Fog computing: Survey of trends, architectures, requirements, and research directions. IEEE Access 6(c), 47980–48009 (2018). https://doi.org/10.1109/ACCESS.2018.2866491

66. Nebbiolo Technologies: Toshiba Digital Solutions Corporation and Nebbiolo Technologies Inc. Sign an Industrial IoT Strategic Partnership Agreement. https://www.ibm.com/blogs/internet-of-things/edge-iot-analytics/ (2018), accessed: 2020-02-27

67. Nebbiolo Technologies Inc.: Fog vs edge computing p. 8 (2016)

68. OpenFog Consortium Architecture Working Group: OpenFog reference architecture for fog computing. Tech. Rep. February (2017). https://doi.org/OPFRA001.020817

69. Pinchuk, A., Sokolov, N., Freinkman, V.: General principles of foggy computing. LastMile (3), 38–45 (2018). https://doi.org/10.22184/2070-8963.2018.72.3.38.45

70. Proferansov, D.Y., Safonova, I.E.: To the question of fog computing and the Internet of Things. Educational Resources and Technology 4(21), 30–39 (2017). https://doi.org/10.21777/2500-2112-2017-4-30-39

71. Puliafito, C., Mingozzi, E., Vallati, C., et al.: Virtualization and migration at the network edge: An overview. In: Proceedings - 2018 IEEE International Conference on Smart Computing, SMARTCOMP 2018. pp. 368–374. IEEE, Taormina, Sicily (2018). https://doi.org/10.1109/SMARTCOMP.2018.00031

72. Puthal, D., Mohanty, S.P., Bhavake, S.A., et al.: Fog computing security challenges and future directions [energy and security]. IEEE Consumer Electronics Magazine 8(3), 92–96 (2019). https://doi.org/10.1109/MCE.2019.2893674

73. Radchenko, G.I., Alaasam, A.B., Tchernykh, A.N.: Comparative analysis of virtualization methods in big data processing. Supercomputing Frontiers and Innovations 6(1), 48–79 (2019). https://doi.org/10.14529/jsfi190107

74. Ravandi, B., Papapanagiotou, I.: A self-learning scheduling in cloud software defined block storage. In: 2017 IEEE 10th International Conference on Cloud Computing, CLOUD. pp. 415–422. IEEE, Honolulu, Hawaii (2017). https://doi.org/10.1109/CLOUD.2017.60

75. Reale, A.: A guide to Edge IoT analytics: Internet of Things blog. https://www.ibm.com/blogs/internet-of-things/edge-iot-analytics/ (2017), accessed: 2020-02-27

76. Russo, G.R.: Model-based auto-scaling of distributed data stream processing applications. In: Middleware 2020 Doctoral Symposium - Proceedings of the 2020 21st International Middleware Conference Doctoral Symposium, Part of Middleware 2020. pp. 5–8. ACM, New York, NY, USA (2020). https://doi.org/10.1145/3429351.3431741

77. Sadashiv, N., Kumar, S.M.: Cluster, grid and cloud computing: A detailed comparison. In: ICCSE 2011 - 6th International Conference on Computer Science and Education, Final Program and Proceedings. pp. 477–482. IEEE, Chennai (2011). https://doi.org/10.1109/ICCSE.2011.6028683

78. Sehgal, N.K., Bhatt, P.C.P., Sehgal, N.K., Bhatt, P.C.P.: Features of private and public clouds. In: Cloud Computing, pp. 51–60. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-77839-6_4

79. Skarlat, O., Karagiannis, V., Rausch, T., et al.: A framework for optimization, service placement, and runtime operation in the fog. In: Proceedings - 11th IEEE/ACM International Conference on Utility and Cloud Computing, UCC 2018. pp. 164–173. IEEE, Zurich (2019). https://doi.org/10.1109/UCC.2018.00025

80. Skarlat, O., Nardelli, M., Schulte, S., et al.: Optimized IoT service placement in the fog. Service Oriented Computing and Applications 11(4), 427–443 (2017). https://doi.org/10.1007/s11761-017-0219-8

81. Skarlat, O., Schulte, S., Borkowski, M., et al.: Resource provisioning for IoT services in the fog. In: Proceedings - 2016 IEEE 9th International Conference on Service-Oriented Computing and Applications, SOCA 2016. pp. 32–39. IEEE, Macau (2016). https://doi.org/10.1109/SOCA.2016.10

82. Smartiply: Edge gateway. https://www.smartiply.com/gateway, accessed: 2020-02-27

83. Smartiply: Mobile platform. https://www.smartiply.com/mobile, accessed: 2020-02-27

84. Sotomayor, B., Montero, R.S., Llorente, I.M., Foster, I.: Virtual infrastructure management in private and hybrid clouds. IEEE Internet Computing 13(5), 14–22 (2009). https://doi.org/10.1109/MIC.2009.119

85. Tran, Q.M., Nguyen, P.H., Tsuchiya, T., Toulouse, M.: Designed features for improving openness, scalability and programmability in the fog computing-based IoT systems. SN Computer Science 1(4), 194 (2020). https://doi.org/10.1007/s42979-020-00197-w

86. Tsai, P.H., Hong, H.J., Cheng, A.C., et al.: Distributed analytics in fog computing platforms using tensorflow and kubernetes. In: 19th Asia-Pacific Network Operations and Management Symposium: Managing a World of Things, APNOMS 2017. pp. 145–150. IEEE (2017). https://doi.org/10.1109/APNOMS.2017.8094194

87. Tseng, F.H., Tsai, M.S., Tseng, C.W., et al.: A lightweight autoscaling mechanism for fog computing in industrial applications. IEEE Transactions on Industrial Informatics 14(10), 4529–4537 (2018). https://doi.org/10.1109/TII.2018.2799230

88. Tuli, S., Basumatary, N., Buyya, R.: EdgeLens: Deep learning based object detection in integrated IoT, fog and cloud computing environments. CoRR abs/1906.11056 (2019), http://arxiv.org/abs/1906.11056

89. Vandenberg, A.: Grid computing for all. In: Guimarães, M. (ed.) Proceedings of the 43nd Annual Southeast Regional Conference, 2005, Kennesaw, Georgia, USA, March 18-20, 2005, Volume 1. p. 3. ACM (2005). https://doi.org/10.1145/1167350.1167353

90. Varshney, P., Simmhan, Y.: Demystifying fog computing: Characterizing architectures, applications and abstractions. In: Proceedings - 2017 IEEE 1st International Conference on Fog and Edge Computing, ICFEC 2017. pp. 115–124. IEEE (2017). https://doi.org/10.1109/ICFEC.2017.20

91. Velasquez, K., Abreu, D.P., Assis, M.R., et al.: Fog orchestration for the Internet of Everything: state-of-the-art and research challenges. Journal of Internet Services and Applications 9(1), 14:1–14:23 (2018). https://doi.org/10.1186/s13174-018-0086-3

92. Wadhwa, H., Aron, R.: Fog computing with the integration of Internet of Things: Architecture, applications and future directions. In: Proceedings - 16th IEEE International Symposium on Parallel and Distributed Processing with Applications, 17th IEEE International Conference on Ubiquitous Computing and Communications, 8th IEEE International Conference on Big Data and Cloud Computing, 11t. pp. 987–994. IEEE, Melbourne (2019). https://doi.org/10.1109/BDCloud.2018.00144

93. Webb, K.: Reviews. Architects of the Information Society: 35 Years of the Laboratory for Computer Science at MIT. Internet Research 10(1), 169–174 (2000). https://doi.org/10.1108/intr.2000.17210aaf.006

94. Weinhardt, C., Anandasivam, A., Blau, B., et al.: Cloud Computing - A Classification, Business Models, and Research Directions. Business & Information Systems Engineering 1(5), 391–399 (2009). https://doi.org/10.1007/s12599-009-0071-2

95. Wen, Z., Yang, R., Garraghan, P., et al.: Fog orchestration for Internet of Things services. IEEE Internet Computing 21(2), 16–24 (2017). https://doi.org/10.1109/MIC.2017.36

96. Yakubu, J., Abdulhamid, S.M., Christopher, H.A., et al.: Security challenges in fog-computing environment: a systematic appraisal of current developments. Journal of Reliable Intelligent Environments 5(4), 209–233 (2019). https://doi.org/10.1007/s40860-019-00081-2

97. Yin, S., Kaynak, O.: Big data for modern industry: Challenges and trends [point of view]. Proc. IEEE 103(2), 143–146 (2015). https://doi.org/10.1109/JPROC.2015.2388958

98. Yousefpour, A., Fung, C., Nguyen, T., et al.: All one needs to know about fog computing and related edge computing paradigms: A complete survey. Journal of Systems Architecture 98, 289–330 (2019). https://doi.org/10.1016/j.sysarc.2019.02.009

99. Zhang, B., Mor, N., Kolb, J., et al.: The cloud is not enough: Saving IoT from the cloud. In: 7th USENIX Workshop on Hot Topics in Storage and File Systems, HotStorage 2015 (2020)

100. Zhang, P., Liu, J.K., Richard Yu, F., et al.: A survey on access control in fog computing. IEEE Communications Magazine 56(2), 144–149 (2018). https://doi.org/10.1109/MCOM.2018.1700333

101. Zheng, W.S., Yen, L.H.: Auto-scaling in Kubernetes-based fog computing platform. In: New Trends in Computer Technologies and Applications, ICS 2018. Communications in Computer and Information Science, vol. 1013, pp. 338–345. Springer (2019). https://doi.org/10.1007/978-981-13-9190-3_35

102. Zhu, J., Chan, D.S., Prabhu, M.S., et al.: Improving web sites performance using edge servers in fog computing architecture. In: Proceedings - 2013 IEEE 7th International Symposium on Service-Oriented System Engineering, SOSE 2013. pp. 320–323. IEEE (2013). https://doi.org/10.1109/SOSE.2013.73