Resource Management in Fog/Edge Computing: A Survey on Architectures, Infrastructure, and Algorithms

Article *in* ACM Computing Surveys · September 2019

DOI: 10.1145/3326066

CITATIONS
178

READS
2,889

2 authors:

Cheol-Ho Hong
Chung-Ang University
59 PUBLICATIONS **516** CITATIONS

SEE PROFILE

Blesson Varghese
University of St Andrews
162 PUBLICATIONS **2,436** CITATIONS

SEE PROFILE

Resource Management in Fog/Edge Computing: A Survey on Architectures, Infrastructure, and Algorithms

CHEOL-HO HONG, Chung-Ang University, South Korea
BLESSON VARGHESE, Queen's University Belfast, United Kingdom

Contrary to using distant and centralized cloud data center resources, employing decentralized resources at the edge of a network for processing data closer to user devices, such as smartphones and tablets, is an upcoming computing paradigm, referred to as fog/edge computing. Fog/edge resources are typically resource-constrained, heterogeneous, and dynamic compared to the cloud, thereby making resource management an important challenge that needs to be addressed. This article reviews publications as early as 1991, with 85% of the publications between 2013–2018, to identify and classify the architectures, infrastructure, and underlying algorithms for managing resources in fog/edge computing.

General Terms: Design, Management, Performance

Additional Key Words and Phrases: fog/edge computing, resource management, architectures, infrastructure, algorithms

1. INTRODUCTION

Accessing remote computing resources offered by cloud data centers has become the de facto model for most Internet-based applications. Typically, data generated by user devices such as smartphones and wearables, or sensors in a smart city or factory are all transferred to geographically distant clouds to be processed and stored. This computing model is not practical for the future because it is likely to increase communication latencies when billions of devices are connected to the Internet [Varghese and Buyya 2018]. Applications will be adversely impacted because of the increase in communication latencies, thereby degrading the overall Quality-of-Service (QoS) and Quality-of-Experience (QoE) [Hong et al. 2018].

An alternative computing model that can alleviate the above problem is bringing computing resources closer to user devices and sensors, and using them for data processing (even if only partial) [Varghese et al. 2016b; Shi et al. 2016]. This would reduce the amount of data sent to the cloud, consequently reducing communication latencies. To realize this computing model, the current research trend is to decentralize some of the computing resources available in large data centers by distributing them towards the edge of the network closer to the end-users and sensors, as depicted in Figure 1. These resources may take the form of either (i) dedicated 'micro' data centers that are conveniently and safely located within public/private infrastructure or (i) Internet nodes, such as routers, gateways, and switches that are augmented with computing capabilities. A computing model that makes use of resources located at the edge of the network is referred to as 'edge computing' [Satyanarayanan et al. 2009; Satyanarayanan 2017]. A model that makes use of both edge resources and the cloud is referred to as 'fog computing' [Bonomi et al. 2012; Dastjerdi and Buyya 2016; Yousefpour et al. 2019].

Contrary to cloud resources, the resources at the edge are: (i) resource constrained - limited computational resources because edge devices have smaller processors and a limited power budget, (ii) heterogeneous - processors with different architectures, and (iii) dynamic - their workloads change, and applications compete for the limited resources. Therefore, managing resources is one of the key challenges in fog and edge computing. The focus of this article is to review the architectures, infrastructure, and algorithms that underpin resource management in fog/edge computing. Figure 2 presents the areas covered by this article.

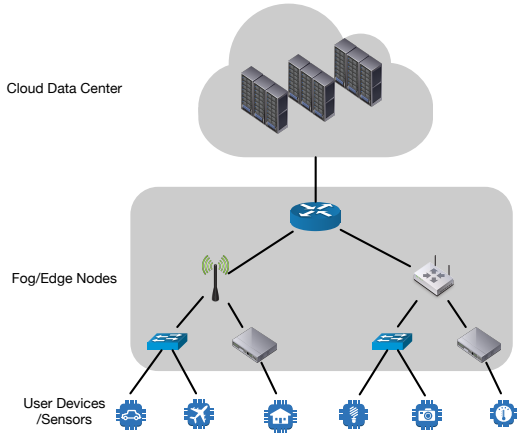

Fig. 1. A fog/edge computing model comprising the cloud, resources at the edge of the network, and end-user devices or sensors

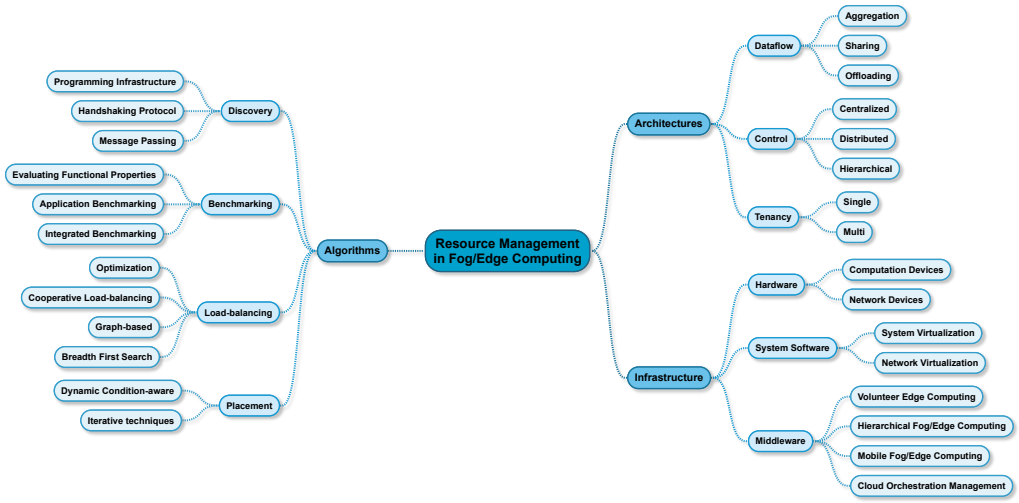

Fig. 2. A classification of the architectures, infrastructure, and algorithms for resource management in fog/edge computing

Figure 3 shows a histogram of the total number of research publications reviewed by this article between 1991 and 2018 under the categories: (i) books and book chapters, (ii) reports, including articles available on pre-print servers or white papers, (iii) conference or workshop papers, and (iv) journal or magazine articles. Similar histograms are provided for each section. More than 85% of the articles reviewed were published from 2013.

The remainder of this article is structured as follows. Section 2 discusses resource management architectures, namely the dataflow, control, and tenancy architectures. Section 3 presents the infrastructure used for managing resources, such as the hardware, system software, and middleware employed. Section 4 highlights the underlying algorithms, such as discovery, benchmarking, load balancing, and placement. Section 5 suggests future directions and concludes the paper.

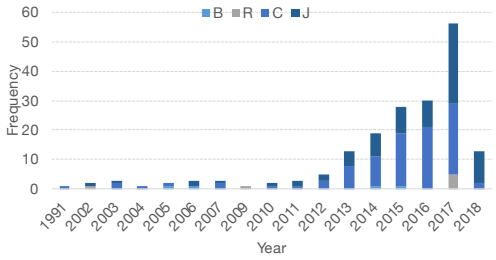

Fig. 3. A histogram of the total number of research publications on resource management in fog/edge computing reviewed by this article. Legend: B - books or book chapters; R - reports, including articles available on pre-print servers or white papers; C - conference or workshop papers; J - journal or magazine articles.

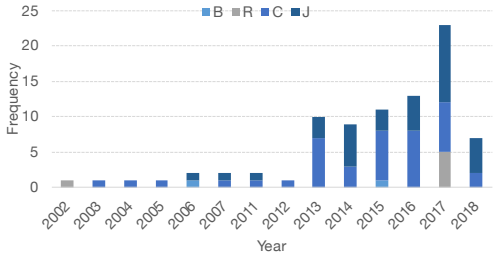

Fig. 4. A histogram of publications reviewed for the classification of architectures for resource management in fog/edge computing. Legend: B - books or book chapters; R - reports, including articles available on pre-print servers or white papers; C - conference or workshop papers; J - journal or magazine articles.

2. ARCHITECTURES

In this survey, the architectures used for resource management in fog/edge computing are classified on the basis of data flow, control, and tenancy.

— *Data flow architectures*: These architectures are based on the direction of movement of workloads and data in the computing ecosystem. For example, workloads could be transferred from the user devices to the edge nodes or alternatively from cloud servers to the edge nodes.
— *Control architectures*: These architectures are based on how the resources are controlled in the computing ecosystem. For example, a single controller or central algorithm may be used for managing a number of edge nodes. Alternatively, a distributed approach may be employed.
— *Tenancy architecture*: These architectures are based on the support provided for hosting multiple entities in the ecosystem. For example, either a single application or multiple applications could be hosted on an edge node.

The survey used 84 research publications to obtain the classification of the architectures shown in the histogram in Figure 4. 86% of publications have been published since 2013.

2.1. Data Flow

This survey identifies key data flow architectures based on how data or workloads are transferred within a fog/edge computing environment. This section considers three data flow architectures, namely aggregation, sharing, and offloading.

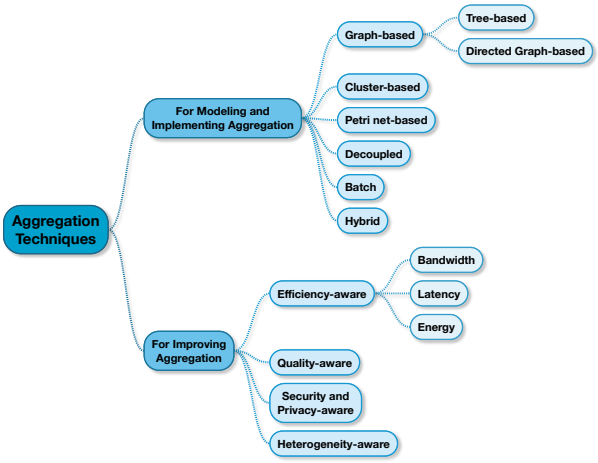

Fig. 5. A classification of aggregation techniques

2.1.1. ***Aggregation.*** In the aggregation model, an edge node obtains data generated from multiple end devices that is then partially computed for pruning or filtering. The aim in the aggregation model is to reduce communication overheads, including preventing unnecessary traffic from being transmitted beyond the edge of the network. Research on aggregation can broadly be classified on the basis of (i) techniques for modeling and implementing aggregation, and (ii) techniques for improving aggregation, as shown in Figure 5.

i. Techniques for Modeling and Implementing Aggregation: The underlying techniques implemented for supporting aggregation have formed an important part of Wireless Sensor Networks (WSNs) [Rajagopalan and Varshney 2006] and distributed data stream processing [de Assunção et al. 2018]. Dense and large-scale sensor networks cannot route all data generated from sensors to a centralized server, but instead need to make use of intermediate nodes along the data path that aggregate data. This is referred to as in-network data aggregation [Fasolo et al. 2007]. We consider WSNs to be predecessors of modern edge computing systems. Existing research in the area of in-network data aggregation can be classified into the following six ways on the basis of the underlying techniques used for modeling and implementing aggregation:

a. Graph-based Techniques: In this survey, we report two graph-based techniques that are used for data aggregation, namely tree-based and directed graph-based techniques.

Tree-based Techniques: Two examples of tree-based techniques are Data Aggregation Trees (DATs) and spatial index trees. DATs are commonly used for aggregation in WSNs using Deterministic Network Models (DNMs) or Probabilistic Network Models (PNMs). Recent research highlights the use of PNMs over DNMs for making realistic assumptions of lossy links in the network by using tree-based techniques for achieving load balancing [He et al. 2014]. Spatial index trees are employed for querying within networks, but have recently been reported for aggregation. EGF is an energy efficient index tree used for both data collection and aggregation [Tang et al. 2013]. This technique is demonstrated to work well when the sensors are unevenly distributed. The sensors are divided into grids, and an index tree is first constructed. Based on the hierarchy, an EGF tree is constructed by merging neighboring grids. Multi-region queries are aggregated in-network and then executed.

Directed Graph-based Techniques: The Dataflow programming model uses a directed graph and is used for WSN applications. Recently, a Distributed Dataflow (DDF) programming model has been proposed in the context of fog computing [Giang et al. 2015]. The model is based on the MQTT protocol, supports the deployment of flow on multiple nodes, and assumes the heterogeneity of devices [Xu et al. 2016].

b. Cluster-based Techniques: These techniques rely on clustering the nodes in the network. For example, energy efficiency could be a key criterion for clustering the nodes. One node from each cluster is then chosen to be a cluster head. The cluster head is responsible for local aggregation in each cluster and for transmitting the aggregated data to another node. Clustering techniques for energy efficient data aggregation have been reported [Jiang et al. 2011]. It has been highlighted that the spatial correlation models of sensor nodes cannot be used accurately in complex networks. Therefore, Data Density Correlation Degree (DDCD) clustering has been proposed [Yuan et al. 2014].

c. Petri Net-based Techniques: In contrast to tree-based techniques, recent research highlights the use of High Level Petri Net (HLPN) referred to as RedEdge for modeling aggregation in edge-based systems [Habib ur Rehman et al. 2017]. Given that fog/edge computing accounts for three layers, namely the cloud, the user device, and the edge layers, techniques that support heterogeneity are required. HLPN facilitates heterogeneity, and the model is validated by verifying satisfiability using an automated solver. The data aggregation strategy was explored for a smart city application and tested for a variety of efficiency metrics, such as latency, power, and memory consumption.

d. Decoupled Techniques: The classic aggregation techniques described above usually exhibit high inaccuracies when data is lost in the network. The path for routing data is determined on the basis of the aggregation technique. However, Synopsis Diffusion (SD) is a technique proposed for decoupling routing from aggregation so that they can be individually optimized to improve accuracy [Nath et al. 2004]. The challenge in SD is that if one of the aggregating nodes is compromised, false aggregations will occur. More recently, there has been research to filter outputs from compromised nodes [Roy et al. 2014]. In more recent edge-based systems, Software-Defined Networking (SDN) is employed to decouple computing from routing [Xu et al. 2016; Zhang et al. 2017]. SDN will be considered in Section 3.2.2.

e. Batch Techniques: This model of aggregation is employed in data stream processing. The data generated from a variety of sources is transmitted to a node where the data is grouped at time intervals to a batch job. Each batch job then gets executed on the node. For example, the underlying techniques of Apache Flink rely on batch processing of incoming data[1]. Similarly, Apache Spark[2] employs the concept of Discretized Streams (or D-Streams) [Zaharia et al. 2013], a micro-batch processing technique that periodically performs batch computations over short time intervals.

f. Hybrid Techniques: These techniques combine one or more of the techniques considered above. For example, the Tributary-Delta approach combines tree-based and Synopsis Diffusion (SD) techniques in different regions of the network [Manjhi et al. 2005]. The aim is to provide low loss rate and present few communication errors while maintaining or improving the overall efficiency of the network.

ii. Techniques for Improving Aggregation: Aggregation can be implemented, such that it optimizes different objectives in the computing environment. These objectives range from communication efficiency in terms of bandwidth, latency, and energy constraints (that are popularly used) to the actual quality of aggregation (or analytics)

[1]https://flink.apache.org/
[2]https://spark.apache.org/

that is performed on the edge node. The following is a classification obtained after surveying existing research on techniques for improving aggregation:

a. Efficiency-aware Techniques: We present three categories of efficiency-aware techniques: the first for optimizing bandwidth, the second for minimizing latency, and the third for reducing energy consumption.

Bandwidth-aware: The Bandwidth Efficient Cluster-based Data Aggregation (BECDA) algorithm has three phases [Mantri et al. 2015]. First, distributed nodes are organized into a number of clusters. Then, each cluster elects a cluster head that aggregates data from within the cluster. Thereafter, each cluster head contributes to intra-cluster aggregation. This approach utilizes bandwidth efficiently for data aggregation in a network and is more efficient than predecessor methods.

Latency-aware: Another important metric that is often considered in edge-based systems for aggregation includes latency [Becchetti et al. 2009; Li et al. 2014]. A mediation architecture has been proposed in the context of data services for reducing latency [Reiff-Marganiec et al. 2014]. In this architecture, policies for filtering data produced by the source based on concepts of complex event processing are proposed. In the experimental model, requests are serviced in near real-time with minimum latency. There is a trade-off against energy efficiency when attempting to minimize latency [Li et al. 2013]. Therefore, techniques to keep latency to a minimum while maintaining constant energy consumption were employed.

Energy-aware: Research in energy efficiency of data aggregation focuses on reducing the power consumption of the network by making individual nodes efficient via hardware and software techniques. For example, in a multi-hop WSN, the energy consumption trade-off with aggregation latency has been explored under the physical interference model [Li et al. 2013]. A successive interference cancellation technique was used, and an energy efficient minimum latency data aggregation algorithm proposed. The algorithm achieves lower bounds of latency while maintaining constant energy. In a mobile device-based edge computing framework, RedEdge, it was observed that the energy consumption for data transfer was minimized [Habib ur Rehman et al. 2017]. However, there is a data processing overhead on the edge node. Energy awareness techniques for edge nodes are an open research area[3].

b. Quality-aware Techniques: Selective forwarding is a technique in which data from end devices are conditionally transmitted to a node for reducing overheads. 'Quality-aware' in this context refers to making dynamic decisions for improving the quality of predictive analytics in selective forwarding [Anagnostopoulos 2014]. In a recent study, the optimal stopping theory was used for maximizing the quality of aggregation without compromising the efficiency of communication [Harth and Anagnostopoulos 2017]. It was noted that instantaneous decision-making that is typically employed in selective forwarding does not account for the historical accuracy of prediction. Quality awareness is brought into this method by proposing optimal vector forwarding models that account for historical quality of prediction.

c. Security-aware Techniques: Aggregation occurring at an edge node between user devices and a public cloud needs to be secure and ensure identity privacy. An Anonymous and Secure Aggregation (ASAS) scheme [Wang et al. 2018] in a fog environment using elliptic curve public-key cryptography, bilinear pairings, and a linearly holomorphic cryptosystem, namely the Castagnos-Laguillaumie cryptosystem [Castagnos and Laguillaumie 2015], has been developed. Another recently proposed technique includes the Lightweight Privacy-preserving Data Aggregation (LPDA) for fog computing [Lu et al. 2017]. LPDA, contrary to ASAS, is underpinned by the homomorphic Paillier encryption, the Chinese Remainder Theorem, and one-way hash chain techniques. Other

[3]http://www.uniserver2020.eu/

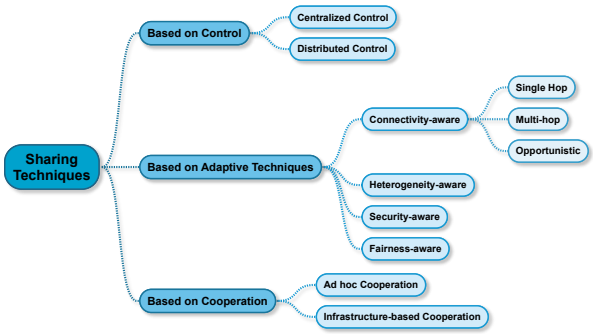

Fig. 6. A classification of sharing techniques

examples of privacy-aware techniques include those employed in fog computing-based vehicle-to-infrastructure data aggregation [Chen et al. 2018].

d. Heterogeneity-aware Techniques: Edge-based environments are inherently heterogeneous [Varghese et al. 2017b]. Heterogeneity of resources here is a reference to different types of fog/edge nodes, including CPU architectures, combination of dedicated micro data centers (CPU-based systems), and traffic routing devices such as routers, base stations, and switches at the edge of the network. Traditional cloud techniques for data aggregation have assumed homogeneous hardware, but there is a need to account for heterogeneity. Some research takes heterogeneous nodes into account for data aggregation in WSNs [Mantri et al. 2013; 2016]. Heterogeneous edge computing is still in infancy [Xiao et al. 2017]. While there is indication of the possibility to use heterogeneous resources in an ideal fog/edge computing model, there is little evidence of such a system that is fully implemented.

2.1.2. ***Sharing***. Contrary to the aggregation model, the sharing model is usually employed when the workload is shared among peers. This model aims at satisfying computing requirements of a workload on a mobile device without offloading it into the cloud, but onto peer devices that are likely to be battery-powered. This results in a more dynamic network given that devices may join and leave the network without notice. Practically feasible techniques proposed for cooperative task execution will need to be inherently energy aware. Research in this area is generally pursued under the umbrella of Mobile Cloud Computing (MCC) [Dinh et al. 2013] and Mobile Edge Computing (MEC) [Mao et al. 2017] and is a successor to peer-to-peer computing [Milojicic et al. 2002].

Research on techniques for sharing can be classified into the following three ways, as shown in Figure 6:

i. Based on Control: Research on control of the sharing model employed in mobile edge devices can be distinguished on the basis of (a) centralized control and (b) distributed control.

a. Centralized Control: In this technique, a centralized controller is used to manage the workload on each edge device in a network. For example, a collection of devices at the edge is modeled as a Directed Acyclic Graph (DAG)-based workflow. The coordination of executing tasks resides with a controller in the cloud [Habak et al. 2015; Habak et al. 2017]. A Software Defined Cooperative Offloading Model (SDCOM) was implemented based on Software Defined Networking (SDN) [Cui et al. 2017]. A controller is placed on a Packet Delivery Network (PDN) gateway that is used to enable cooperation between mobile devices connected to the controller. The controller aims at

reducing traffic on the gateway and ensuring fairness in energy consumption between mobile devices. To deal with dynamically arriving tasks, an Online Task Scheduling (OTS) algorithm was developed.

Centralized techniques are fairly common in the literature since they are easier to implement. However, they suffer from scalability and single point failures as is common in most centralized systems.

b. Distributed Control: In the area of distributed control among edge devices, there seems to be relatively limited research. A game theoretic approach was employed as a decentralized approach for achieving the Nash equilibrium among cooperative devices [Chen 2015]. The concept of the Nash equilibrium in the sharing model is taken further to develope the Multi-item Auction (MIA) model and Congestion Game (COG)-based sharing [Ma et al. 2015].

ii. Based on Adaptive Techniques: These techniques are nature inspired and solve multi-objective optimization problems [Kimovski et al. 2018]. There are different objectives in a system that employs a sharing model. For example, the sharing model at the edge can be employed in a battlefield scenario [Gao 2014]. In this context, latencies need to be minimum, and the energy consumption of the devices needs to be at optimum. Based on existing research, the following adaptive techniques are considered:

a. Connectivity-aware: The sharing model needs to know the connectivity between devices, for example, in the above battlefield scenario. A mobile device augments its computing when peer devices come within its communication range [Gao 2014]. Then a probabilistic model predicts whether a task potentially scheduled on a peer device can complete execution in time when it is in the coverage of the device. Connectivity-aware techniques can be single hop, multi-hop, or opportunistic [Mtibaa et al. 2013].

Single Hop Techniques: In this technique, a device receives a list of its neighbors that form a fully connected network. When a workload is shared by a device, the workload will be distributed to other devices that are directly connected to the device [Mtibaa et al. 2013].

Multi-hop Techniques: Each device computes the shortest path to every other node in the network that can share its workload. The work is usually shared with devices that may reduce the overall energy footprint. The benefit of a multi-hop technique in the sharing model compared to single hop techniques is that a larger pool of resources can be tapped into for more computationally intensive workloads. A task distribution approach using a greedy algorithm to reduce the overall execution time of a distributed workload was recently proposed [Funai et al. 2016].

Opportunistic Techniques: The device that needs to share its workload in these techniques checks whether its peers can execute a task when it is within the communication range. This is predicted via contextual profiling or historical data of how long a device was within the communication range of its peers. In recent research, a connectivity-aware opportunistic approach was designed such that: (i) data and code for the job can be delivered in a timely manner, (ii) sequential jobs are executed on the same device so that intermediate data does not have to be sent across the network, and (iii) there is distributed control, and jobs are loosely coupled [Shi et al. 2012]. The jobs are represented as a Directed Acyclic Graph (DAG), and the smallest component of a job is called a PNP-block that is used as the unit scheduled onto a device. In the context of Internet-of-Things (IoT) for data-centric services, it is proposed that a collection of mobile devices forms a mobile cloud via opportunistic networking to service the requests of multiple IoT sensors [Borgia et al. 2016].

b. Heterogeneity-aware: Edge devices in a mobile cloud are heterogeneous at all levels. Therefore, the processor architecture, operating system, and workload deployment pose several challenges in facilitating cooperation [Sanaei et al. 2014]. There is research that assumes that the architectures of the cooperating edge are similar, but

have different energy and memory or system utilization requirements. These parameters are used for coded computation [Keshtkarjahromi et al. 2018]. There is recent research tackling heterogeneity-related issues in mobile networks. For example, a work sharing approach named Honeybee was proposed in which cycles of heterogeneous mobile devices are used to serve the workload from a given device [Fernando et al. 2016]. The approach accounts for devices leaving/joining the system. Similarly, a framework based on service-oriented utility functions was proposed for managing heterogeneous resources that share tasks [Nishio et al. 2013]. A resource coordinator delegates tasks to resources in the network so that parameters, such as gain and energy, are optimized using convex optimization techniques.

c. Security-aware: A technique to identify and isolate malicious attacks that could exist in a device used in the sharing model, referred to as HoneyBot, has been proposed [Mtibaa et al. 2015]. A few of the devices in a mobile network are chosen as HoneyBots for monitoring malicious behavior. In the provided experimental results, a malicious device can be identified in 20 minutes. Once a device is identified to be malicious, it is isolated from the network to keep the network safe.

d. Fairness-aware: Fairness has been defined as a multi-objective optimization problem. The objectives are to reduce the drain on the battery of mobile devices so as to prolong the network lifetime, and at the same time improve the performance gain of the workload shared between devices [Viswanathan et al. 2016]. The processing chain of mobile applications was modeled as a DAG and assumed that each node of the DAG is an embarrassingly parallel task. Each task was considered as a Multi-objective Combinatorial Bottleneck Problem (M-CBP) solved using a heuristic technique.

iii. Based on Cooperation: Edge devices can share workloads (a) either in a less defined environment that is based on ad hoc cooperation, or (b) in a more tightly coupled environment where there is infrastructure to facilitate cooperation.

a. Ad Hoc Cooperation: Setting up ad hoc networks for device-to-device communication is not a new area of research. Ad hoc cooperation has been reported for MCC in the context of the sharing model for the edge [Panneerselvam et al. 2016]. There is recent research that has coined the term "transient clouds," in which neighboring mobile devices form an ad hoc cloud and the underlying task management algorithm is based on a variant of the Hungarian method [Penner et al. 2014].

b. Infrastructure-based Cooperation: There is research on the federation of devices at the edge of the network to facilitate cooperation [Farris et al. 2017]. This results in more tightly coupled coalitions than ad hoc clouds, and more cost effectiveness than dedicated micro cloud deployment.

2.1.3. ***Offloading.*** Offloading is a technique in which a server, an application, and the associated data are moved on to the edge of the network. This either augments the computing requirements of individual or a collection of user devices, or brings services in the cloud that process requests from devices closer to the source. Research in offloading can be differentiated in the following two ways, as presented in Figure 7:

i. Offloading from User Device to Edge: This technique augments computing in user devices by making use of edge nodes (usually a single hop away). The two main techniques used are application partitioning and caching mechanisms.

a. Application Partitioning: One example of offloading from devices to the edge via application partitioning is in the GigaSight architecture in which Cloudlet VMs [Satyanarayanan et al. 2009] are used to process videos streamed from multiple mobile devices [Simoens et al. 2013]. The Cloudlet VM is used for denaturing, a process of removing user-specific content for preserving privacy. The architecture employed is presented as a Content Delivering Network (CDN) in reverse. In this survey, we discuss the following four approaches and three models used for application partitioning.

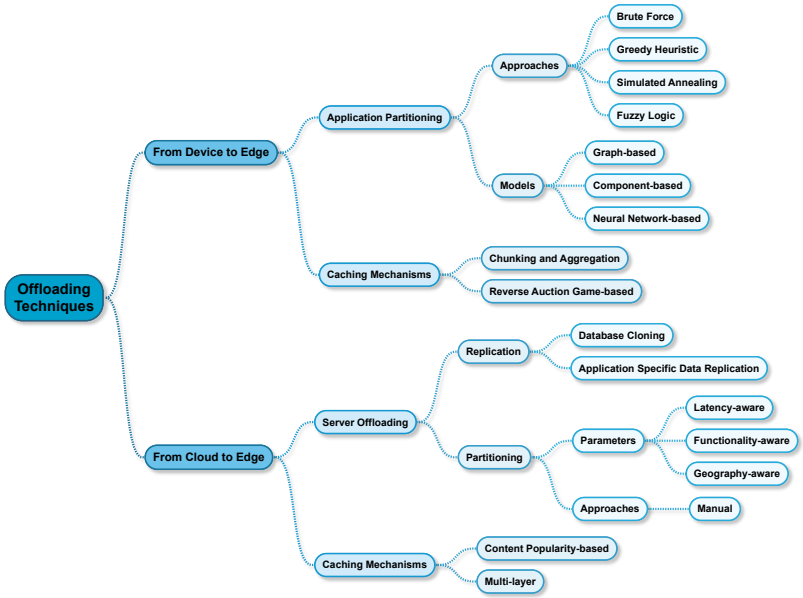

Fig. 7. A classification of offloading techniques

Approaches: Four approaches are considered, namely, brute force, greedy heuristic, simulated annealing, and fuzzy logic.

Brute Force: There is a study under the umbrella of ENGINE that proposes an exhaustive brute force approach, in which all possible combinations of offloading plans (taking the cloud, edge nodes, and user devices) are explored [Chen et al. 2017]. The plan with the minimum execution time for a task is then chosen. This approach simply is not a practical solution given the time needed to derive a plan, but instead could provide insight into the search space.

Greedy Heuristic: ENGINE also incorporates a greedy approach that focuses on merely minimizing the time taken for completing the execution of a task on the mobile device [Chen et al. 2017]. An offloading plan is initially generated for each task on a mobile device, and then iteratively refined to keep the total monetary costs low. Similarly, FogTorch, a prototype implementation of an offloading framework, uses a greedy heuristic approach for deriving offloading plans [Brogi and Forti 2017].

Simulated Annealing: Another approach is simulated annealing, in which the search space is based on the utilization of fog and cloud nodes, total costs, and the completion time of an application to obtain an offloading plan that minimizes the costs and the completion time of the task [Chen et al. 2017].

Fuzzy Logic: There is research highlighting that an application from a user device can be partitioned and placed on fog nodes using fuzzy logic [Mahmud et al. 2018]. The goal is to improve the Quality-of-Experience (QoE) measured by multiple parameters such as service access rate, resource requirements, and sensitivity toward data processing delay. Fuzzy logic is used to prioritize each application placement request by considering the computational capabilities of the node.

Models: The three underlying models used for application partitioning from devices to the edge are graph-based, component-based, and neural network-based.

Graph-based: CloneCloud employs a graph-based model for the automated partitioning of an application [Chun et al. 2011]. Applications running on a mobile device are partitioned and then offloaded onto device clones in the cloud. In the run-time, this concept translates to migrating the application thread onto the clone, after which it is brought back onto the original mobile device. Similarly, in another graph-based approach, each mobile application task to be partitioned is represented as a Directed Acyclic Graph (DAG) [Bhattcharya and De 2016]. The model assumes that the execution time, migration time, and data that need to be migrated for each task are known a priori via profiling. Aspect-oriented programming is then used to obtain traces of sample benchmarks. Thereafter, a trace simulation is used to determine whether offloading to the edge nodes would reduce execution time.

Component-based: In this case, the functionalities of an application (a web browser) that runs on a device are modeled as components that are partitioned between the edge server and the device [Takahashi et al. 2015]. The example demonstrated is Edge Accelerated Web Browsing (EAB), in which individual components of a browser are partitioned across the edge and the device. The contents of a web page are fetched and evaluated on the edge while the EAB client merely displays the output.

Neural Network-based: Recent research highlights the distribution of deep neural networks across user devices, edge nodes, and the cloud [Kang et al. 2017; Teerapittayanon et al. 2017]. The obvious benefit is that the latency of inferring from a deep neural network is reduced for latency-critical applications without the need to transmit images/video far from the source. Deep networks typically have multiple layers that can be distributed over different nodes. The Neurosurgeon framework models the partitioning between layers that will be latency- and energy-efficient from end-to-end [Kang et al. 2017]. The framework predicts the energy consumption at different points of partitioning in the network and chooses a partition that minimizes data transfer and consumes the least energy. This research was extended towards distributing neural networks across geographically distributed edge nodes [Teerapittayanon et al. 2017].

b. Caching Mechanisms: This is an alternative to application offloading. In this mechanism, a global cache is made available on an edge node that acts as a shared memory for multiple devices that need to interact. This survey identifies two such mechanisms, namely chunking and aggregation, and a reverse auction game-based mechanism.

Chunking and Aggregation: The multi Radio Access Technology (multi-RAT) was proposed as an architecture for upload caching. In this model, VMs are located at the edge of the network, and a user device uploads chunks of a large file onto them in parallel [Tokunaga et al. 2016]. Thereafter, an Aggregation VM combines these chunks that are then moved onto a cloud server.

Reverse Auction Game-based: An alternate caching mechanism based on cooperation of edge nodes was proposed in [Xu et al. 2018]. The users generate videos that are shared between the users via edge caching. The mechanism uses a reverse auction game to incentivize caching.

ii. Offloading from the Cloud to the Edge: The direction of data flow is opposite that considered above; in this case, a workload is moved from the cloud to the edge. There are three techniques that are identified in this survey including server offloading, caching mechanisms, and web programming.

a. Server Offloading: A server that executes on the cloud is offloaded to the edge via either replication or partitioning. The former is a naive approach that assumes that a server on the cloud can be replicated on the edge.

Replication: Database cloning and application data replication are considered [Lin et al. 2007; Gao et al. 2003].

Database Cloning: The database of an application may be replicated at the edge of the network and can be shared by different applications or users [Lin et al. 2007].

Application-specific Data Replication: In contrast to database cloning, a specific application may choose to bring data relevant to the users to the edge for the seamless execution of the application [Gao et al. 2003]. However, both database cloning and application-specific data replication assume that edge nodes are not storage-limited, so they may not be feasible in resource-constrained edge environments.

Partitioning: We now consider the server partitioning parameters that are taken into account in offloading from the cloud to the edge. The parameters considered in partitioning are functionality-aware, geography-aware, and latency-aware.

Functionality-aware: Cognitive assistance applications, for example Google Glass, are latency-critical applications, and the processing required for these applications cannot be provided by the cloud alone. Therefore, there is research on offloading the required computation onto Cloudlet VMs to meet the processing and latency demands of cognitive assistance applications [Chen et al. 2015]. The Gabriel platform built on OpenStack++ is employed for VM management via a control VM, and for deploying image/face recognition functionalities using a cognitive VM on Cloudlet.

Geography-aware: The service requests of online games, such as PokeMon Go, are typically transmitted from user devices to a cloud server. Instead of sending traffic to data centers, the ENORM framework partitions the game server and deploys it on an edge node [Wang et al. 2017c]. Geographical data relevant to a specific location is then made available on an edge node. Users from the relevant geographical region connect to the edge node and are serviced as if they were connected to the data center. ENORM proposes an auto-scaling mechanism to manage the workload for maximizing the performance of containers that are hosted on the edge by periodically monitoring resource utilization.

Latency-aware: Similar to ENORM, a study by Báguena et al. aimed at partitioning the back-end of an application logic traditionally located on clouds so as to service application requests in real-time [Báguena et al. 2016]. In the proposed hybrid edge-assisted execution model for LTE networks, application requests are serviced by both the cloud and the edge networks based on latency requirements. This differs from the ENORM framework, in which the server is partitioned along geographical requirements.

b. Caching Mechanisms: Content popularity and multi-layer caching are identified.

Content Popularity-based: Content-Delivery Networks (CDNs) and ISP-based caching are techniques employed to alleviate congestion in the network when downloading apps on user devices. However, there are significant challenges arising from the growing number of devices and apps. A study by Bhardwaj et al. presented the concept of caching mechanisms specific to apps on edge nodes, such as routers and small cells, referred to as eBoxes [Bhardwaj et al. 2015]. This concept is called AppSachets and employs two caching strategies: based on popularity and based on the cost of caching. The research was validated on Internet traffic originating from all users at the Georgia Institute of Technology for a period of 3 months.

Similarly, there is research aimed at caching data at base stations that will be employed in 5G networks [Zeydan et al. 2016]. To achieve this, traffic is monitored to estimate content popularity using a Hadoop cluster. Based on the estimate, content is proactively cached at a base station.

Multi-layer Caching: Multi-layer caching is a technique used in content delivery for Wireless Sensor Networks (WSNs) [Vu et al. 2017]. The model assumes that a global cache is available at a base station that can cache data from data centers, and that localized caches are available on edge nodes. Two strategies are employed in this technique. The first is uncoded caching, in which each node is oblivious of the cache

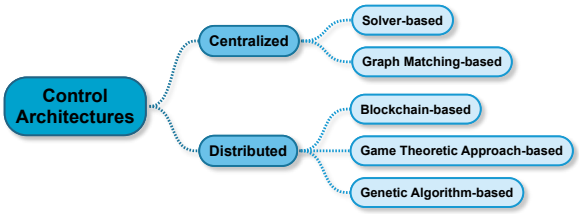

Fig. 8. A classification of control architectures for resource management in fog/edge computing

content of other nodes, and therefore no coordination of data is required. The second technique is coded caching, in which the cached content is coded such that all edge nodes are required to encode the content for the users.

Other miscellaneous techniques are used to support offloading from the cloud to the edge. These are application specific, and are determined by the way the application is programmed. For example, there is research highlighting the use of web programming that makes use of the client-edge-server architecture, such that some component of the client executes in edge nodes. The Spaceify ecosystem enables the execution of Spacelets on edge nodes that are embedded JavaScripts that use the edge nodes to execute tasks to service user requests [Savolainen et al. 2013]. An indoor navigation use-case is demonstrated for validating the Spaceify concept.

2.2. Control

A second method for classifying architectures for resource management in fog/edge environments is based on control of the resources. This survey identifies two such architectures, namely centralized and distributed control architectures, as shown in Figure 8. Centralized control refers to the use of a single controller that makes decisions on the computations, networks, or communication of the edge resources. On the contrary, when decision-making is distributed across the edge nodes, we refer to the architecture as distributed. This section extends the discussion on control techniques that was previously presented on sharing techniques in the survey.

2.2.1. ***Centralized.*** There is a lot of research on centralized architectures, but we identify two centralized architectures, namely, (i) solver-based, and (ii) graph matching-based.

i. Solver-based: Mathematical solvers are commonly used for generating deployment and redeployment plans for scheduling workloads in grids, clusters, and clouds. Similar approaches have been adopted for edge environments. For example, a Least Processing Cost First (LPCF) method was proposed for managing task allocation in edge nodes [Mohan and Kangasharju 2016]. The method is underpinned by a solver aimed at minimizing processing costs and optimizing network costs. The solver is executed on a centralized controller for generating the assignment plan.

ii. Graph Matching-based: An offloading framework that accounts for device-to-device and cloud offloading techniques was proposed [Chen and Zhang 2017]. Tasks were offloaded via a three-layer graph-matching algorithm that is first constructed by taking the offloading space (mobiles, edge nodes, and the cloud) into account. The problem of minimizing the execution time of the entire task is mapped onto the minimum weight-matching problem in the three-layer graph. A centralized technique using the Blossom algorithm was used to generate a plan for offloading.

2.2.2. ***Distributed.*** Three distributed architectures are identified: (i) blockchain-based, (ii) game theoretic-based, and (iii) genetic algorithm-based.

i. Blockchain-based: Blockchain technology is used as an underpinning technique for implementing distributed control in edge computing systems [Stanciu 2017]. The technique is built on the IEC 61499 standard that is a generic standard for distributed control systems. In this model, Function Blocks, an abstraction of the process, was used as an atomic unit of execution. Blockchains make it possible to create a distributed peer-to-peer network without having intermediaries, and therefore naturally lend themselves to the edge computing model in which nodes at the edge of the network can communicate without mediators. The Hyperledger Fabric, a distributed ledger platform used for running and enforcing user-defined smart contracts securely, was used.

ii. Game Theoretic Approach-based: The game theoretic approach is used for achieving distributed control for offloading tasks in the multi-channel wireless interference environment of mobile-edge cloud computing [Chen et al. 2016]. It was demonstrated that finding an optimal solution via centralized methods is NP-hard. Therefore, the game theoretic approach is very suitable in such environments. The Nash equilibrium was achieved for distributed offloading, while two metrics, namely the number of benefitting cloud users and the system-wide computational overhead, were explored to validate the feasibility of the game theoretic approach over centralized methods.

iii. Genetic Algorithm-based: Typically, in IoT-based systems, the end devices are sensors that send data over a network to a computing node that makes all the decision regarding all aspects of networking, communication, and computation. The Edge Mesh approach aims at distributing decision-making across different edge nodes [Sahni et al. 2017]. For this purpose, Edge Mesh uses a computation overlay network along with a genetic algorithm to map a task graph onto the communication network to minimize energy consumption. The variables considered in the genetic algorithm are the Generation Gap used for crossover operations, mutation rate, and population size.

Additionally, there are other upcoming concepts, such as sensor function virtualization (SFV), which can support distributed decision-making. SFV modularizes and deploys sensor functions anywhere in an IoT network [Van den Abeele et al. 2015]. The advantage of the SFV technique is that modules can be added at runtime on multiple nodes. SFV as a concept is still in infancy and needs to be demonstrated in a real world IoT testbed.

2.3. Tenancy

A third method for classifying architectures for resource management in fog/edge environments is tenancy. The term tenancy in distributed systems refers to whether or not underlying hardware resources are shared between multiple entities for optimizing resource utilization and energy efficiency. A single-tenant system refers to the exclusive use of the hardware by an entity. Conversely, a multi-tenant system refers to multiple entities sharing the same resource. An ideal distributed system that is publicly accessible needs to be multi-tenant.

The OpenFog reference architecture highlights multi-tenancy as an essential feature in fog/edge computing [Consortium 2017]. An application server may be offloaded from the cloud to the edge and service users. Therefore, the entities that share the hardware resources in this context are the applications that are hosted on the edge, and the users that are serviced by the edge server.

In this article, we propose a classification of tenancy in fog/edge computing in two dimensions - applications and users. As shown in Figure 9, the followings are the four possibilities in the taxonomy:

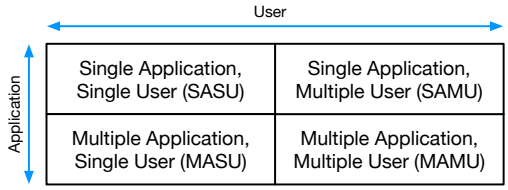

Fig. 9. A taxonomy of tenancy-based architectures for resource management in Fog/Edge computing.

i. *Single Application, Single User (SASU)*: The edge node executes one application, and only the user can connect to the application. The application and the user solely use the hardware resources. The infrastructure is likely to be a private experimental test-bed.
ii. *Single Application, Multiple User (SAMU)*: The edge node executes one application that supports multiple users. Although the underlying hardware resources are not shared among applications, there is a higher degree of sharing than SASU since multiple user requests are serviced by the edge node.
iii. *Multiple Application, Single User (MASU)*: The edge node hosts multiple applications, but each application can only support a single user. This form of tenancy may be used for experimental purposes (or stress-testing the system) during the development of an ideal infrastructure.
iv. *Multiple Application, Multiple User (MAMU)*: The edge node hosts multiple applications, and many users can connect to an individual application. This is an ideal infrastructure and is representative of a publicly accessible infrastructure.

There are two techniques that support multi-tenancy, namely, system virtualization and network slicing.

1) *System Virtualization*: At the system level, virtualization is a technique employed to support multi-tenancy. A variety of virtualization technologies are currently available such as traditional virtual machines (VMs) and containers (considered in Section 3.2.1). VMs have a larger resource footprint than containers. Therefore, lightweight virtualization currently utilized in edge computing incorporates the latter [Wang et al. 2017c; Liu et al. 2016; Morabito et al. 2018]. Virtualization makes it possible to isolate resources for individual applications, whereby users can access applications hosted in a virtualized environment. For example, different containers of multiple applications may be concurrently hosted on an edge node.

2) *Network Slicing*: At the network level, multiple logical networks can be run on top of the physical network, so that different entities with different latency and throughput requirements may communicate across the same physical network [Sallent et al. 2017]. The key principles of Software Defined Networking (SDN) and Network Functions Virtualization (NFV) form the basis of slicing (considered in Section 3.2.2). The ongoing European project SESAME[4] (Small cells coordination for Multi-tenancy and Edge services) tackles the challenges posed by network slicing. The network bandwidth may also be partitioned across tenants, and also referred to as slicing. EyeQ is a framework that supports fine-grained control of network bandwidth for edge-based applications [Jeyakumar et al. 2013]. The framework provides end-to-end minimum bandwidth guarantees, thereby providing an efficient implementation for network performance isolation at the edge.

[4]http://www.sesame-h2020-5g-ppp.eu/Home.aspx

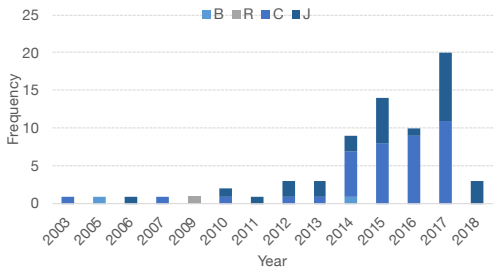

Fig. 10. A histogram of publications reviewed for the classification of the infrastructure for resource management in fog/edge computing. Legend: B - books or book chapters; R - reports, including articles available on pre-print servers or white papers; C - conference or workshop papers; J - journal or magazine articles.

3. INFRASTRUCTURE

The infrastructure for fog/edge computing provides facilities comprising hardware and software to manage the computation, network, and storage resources [Confais et al. 2017a] for applications utilizing the fog/edge. In this article, the infrastructure for resource management in fog/edge computing is classified into the following three categories:

— *Hardware*: Recent studies in fog/edge computing suggest exploiting small-form-factor devices such as network gateways, WiFi Access Points (APs), set-top boxes, small home servers, edge ISP servers, cars, and even drones as compute servers for resource efficiency [Stojmenovic 2014]. Recently, these devices are being equipped with single-board computers (SBCs) that offer considerable computing capabilities. Fog/edge computing also utilizes commodity products such as desktops, laptops, and smartphones.
— *System software*: System software runs directly on fog/edge hardware resources such as the CPU, memory, and network devices. It manages resources and distributes them to the fog/edge applications. Examples of system software include operating systems and virtualization software.
— *Middleware*: Middleware runs on an operating system and provides complementary services that are not supported by the system software. The middleware coordinates distributed compute nodes and performs deployment of virtual machines or containers to each fog/edge node.

This section reviewed 70 research publications to obtain the classification of the infrastructure shown in the histogram in Figure 10. 84% of publications were published since 2013.

3.1. Hardware

Fog/edge computing forms a computing environment that uses low-power mobile devices, home gateways, home servers, edge ISP servers, and routers. These small-form-factor devices nowadays have competent computing capabilities and are connected to the network. The combination of these small compute servers enables a cloud computing environment that can be leveraged by a rich set of applications processing Internet of Things (IoT) and cyber-physical systems (CPS) data. Hardware used for fog/edge computing can be classified in two ways as shown in Figure 11.

3.1.1. Computation Devices. Computation devices for the fog/edge include single-board computers and commodity products that are designed for processing fog/edge data.

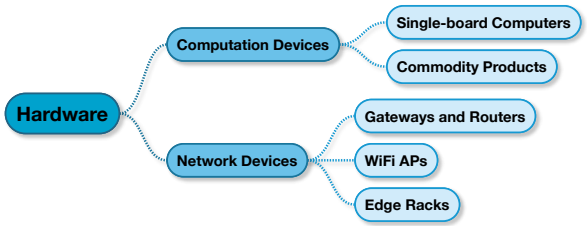

Fig. 11. A classification of hardware

i. Single-board Computers: Single-board computers (SBC) such as Raspberry Pi are often used as fog/edge nodes [Johnston et al. 2018; Amento et al. 2016; Bellavista and Zanni 2017]. An SBC is a small computer based on a single circuit board integrating a CPU, memory, network, and storage devices, and other components together. The small computer does not have expansion slots for peripheral devices. FocusStack [Amento et al. 2016] uses multiple Raspberry Pi boards installed in connected vehicles and drones to build a cloud system. FocusStack deploys a video sharing application where cameras in cars and drones capture moving scenes, and the Raspberry Pi boards process and share them. Bellavista et al. [Bellavista and Zanni 2017] used Raspberry Pi for IoT gateways that are close to sensors and actuators and therefore enable efficient data aggregation. Hong et al. [Hong 2017] utilized Raspberry Pi for crowd-sourced fog computing and programmable IoT analytics.

ii. Commodity Products: Commodity products such as desktops, laptops, and smartphones have been utilized as fog/edge nodes as well. For example, a recent study [Hong et al. 2016] attempted to build a cloud computing environment with laptops and smartphones used in classrooms, movie theaters, and cafes. As the owners of these devices do not always fully utilize the computational resources, fog computing providers may purchase the devices for reselling idle resources to other users. Hong et al. [Hong et al. 2016] developed an animation rendering service using under-utilized laptops in fog computing that offers cost-effectiveness compared to services in traditional cloud computing.

3.1.2. Network Devices. Network devices for fog/edge computing consist of gateways, routers, WiFi APs, and edge racks that are located in the edge and mainly process network traffics.

i. Gateways and Routers: Network gateways and routers are potential devices for fog/edge computing because they establish a data path between end users and network providers. Aazam et al. adopted a common gateway to decide whether the received data from IoT devices would be sent to data center clouds [Aazam and Huh 2014]. Such smart gateways help in better utilization of network bandwidth.

ii. WiFi APs: ParaDrop [Liu et al. 2016], an edge computing framework, exploits the fact that WiFi APs or other wireless gateways are ubiquitous and always turned on.

iii. Edge Racks: Global Environment for Network Innovations (GENI) packs network, computing, and storage resources into a single rack [Gosain et al. 2016]. GENI implements an edge computing environment by deploying GENI racks at several networked sites. These racks currently connect over 50 sites in the USA and are used as Future Internet and Distributed Cloud (FIDC) testbeds.

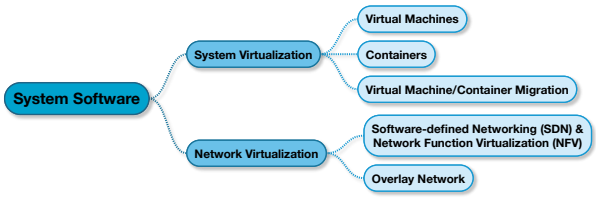

Fig. 12. A classification of system software

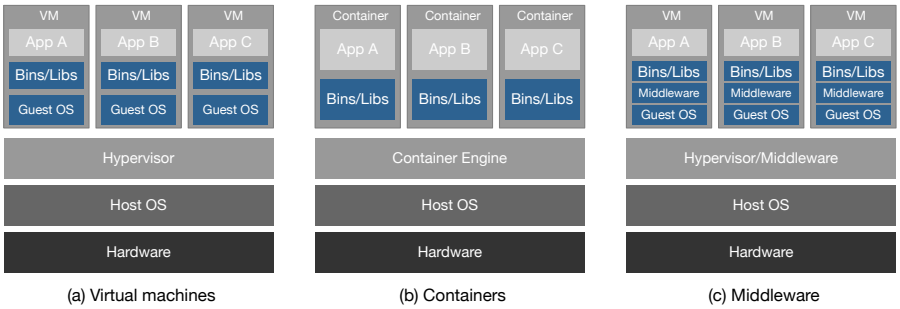

Fig. 13. Architectures of virtual machines, containers, and middleware for resource management in fog/edge computing

3.2. System Software

System software for the fog/edge is a platform designed to operate directly on fog/edge devices and manage the computation, network, and storage resources of the devices. Examples include virtual machines (VMs) and containers. The system software needs to support multi-tenancy and isolation because fog/edge computing accommodates several applications from different tenants. System software used for fog/edge computing can be classified into two categories, system virtualization and network virtualization, as shown in Figure 12.

3.2.1. System Virtualization. System virtualization allows multiple operating systems to run on a single physical machine. System virtualization enables fault and performance isolation between multiple tenants in the fog/edge. It partitions resources for each tenant so that one tenant cannot access other tenants' resources. The fault of a tenant, therefore, cannot affect other tenants. System virtualization also limits and accounts for the resource usage of each tenant so that a tenant cannot monopolize all the available resources in the system. In particular, system virtualization is only an enabling technology for fog computing, as fog computing accommodates multiple tenants at the edge of a network. This section deals with traditional virtual machines, recent containers, and VM/container migration software for supporting system virtualization.

i. Virtual Machines: A Virtual Machine (VM) is a set of virtualized resources used to emulate a physical computer. Virtualized resources include CPUs, memory, network, storage devices [Chiueh and Brook 2005], and even GPUs and FPGAs [Hong et al. 2017b]. Virtualization software called a hypervisor (e.g., Xen [Barham et al. 2003] or KVM [Kivity et al. 2007]) virtualizes the physical resources and provides the virtualized resources in the form of a VM. The tenant installs an operating system and runs applications in the VM, regarding the VM as a real physical machine. The VM architecture is shown in the left side of Figure 13. Virtualization isolates the execution

environment between fog/edge tenants. Each tenant maintains its own VMs, and IoT and CPS devices of the tenant send data to the VMs for processing and storage [Satyanarayanan et al. 2009; Gu et al. 2017; Wang et al. 2017a; Gosain et al. 2016].

Cloudlet provides an early form of fog computing by offering resource-rich VMs for mobile devices in close proximity [Satyanarayanan et al. 2009]. As a mobile device connects to a VM over a wireless LAN, Cloudlets achieve low latency when the mobile device offloads tasks to the VM.

Gu et al. [Gu et al. 2017] proposed a fog computing architecture using VMs for a medical cyber-physical system (MCPS) [Lee et al. 2012]. To receive fast and accurate medical feedbacks, the MCPS system utilizes computational resources close to medical devices. The research utilizes low power sensors and actuators for collecting health information and then sends the collected information to a VM in the network edge (e.g., base stations) for storage and analyses. The research associates several medical devices of a tenant with a VM running in the edge.

Wang et al. [Wang et al. 2017a] implemented a real time surveillance infrastructure where surveillance cameras send images to a distributed edge cloud platform. The surveillance system launches a group of VMs to which surveillance tasks are distributed. When the load is high across the cloud, the system elastically launches new VMs to secure more computational power and network bandwidth.

GENI [Gosain et al. 2016] provides GENI racks to realize an edge-based cloud computing platform for university campuses. Each rack consists of Layer-2 and -3 switches and compute nodes that provide VMs to university students on demand. This infrastructure is available around 50 campuses in the USA.

ii. Containers: Containers are an emerging technology for cloud computing that provides process-level lightweight virtualization [Vaughan-Nichols 2006; Haydel et al. 2015]. Containers are multiplexed by a single Linux kernel, so that they do not require an additional virtualization layer compared to virtual machines. Although they share the same OS kernel, they still offer operating systems virtualization principles [Pahl and Lee 2015] where each user is given an isolated environment for running applications. The architecture of containers is shown in the middle of Figure 13.

Namespaces in Linux provide containers with their own view of the system, and *cgroups* are responsible for resource management such as CPU allocation to containers. This lightweight virtualization allows containers to start and stop rapidly and to achieve performance similar to that of the native environment. In addition, containers are usually deployed with a pre-built application and its dependent libraries, focusing on Platform-as-a-Service (PaaS) that makes container-based applications easily deployed and orchestrated. Representative container tools include LXC [Helsley 2009] and Docker [Merkel 2014] for building and deploying containers, and Kubernetes [Brewer 2015] for orchestration.

Lightweight virtualization implemented by containers facilitates the adoption of performance-limited resources in fog/edge computing nodes [Bellavista and Zanni 2017; Morabito and Beijar 2016; Liu et al. 2016]. Bellavista et al. [Bellavista and Zanni 2017] employed Docker-based containers on a Raspberry Pi 1 board that is used as a fog node for collecting data from heterogeneous sensors in a transit vehicle or other infrastructural components. Morabito et al. [Morabito and Beijar 2016] utilized single-board computers, including RaspberryPi 2, Odroid C1+, and Odroid XU4 boards as edge processing devices running Docker containers. ParaDrop [Liu et al. 2016] adopted lightweight containerization for WiFi Access Points (APs) or other wireless gateways.

Containers provide feasible performance for fog/edge computing with performance-limited resources. Morabito et al. [Morabito and Beijar 2016; Morabito 2017] showed that the container engine only incurs a CPU overhead of approximately 2% in the worst case compared with the native environment. They employed various powerful

embedded boards that equip recent ARM processors. Kaur et al. [Kaur et al. 2017] claimed that containers do not impose significant overheads on the CPU and memory utilization and network bandwidth based on their evaluations.

iii. Virtual Machine/Container Migration: Virtual Machine (VM) or container migration moves a running VM or container to different physical machines for load-balancing and fault tolerance [Medina and García 2014; Ahmad et al. 2015; Forsman et al. 2015; Osanaiye et al. 2017]. VM/container migration approaches can be categorized into three classes [Osanaiye et al. 2017]: cold migration, hot migration, and live migration. Cold migration shuts down the VM/container before migration and restarts it on a different machine. Hot migration suspends the VM/container before migration and resumes it later rather than shutting down it. Hot migration does not affect the applications running on the VM or container as the applications are not restarted. Live migration allows the applications to run continuously during migration as the VM/container is seamlessly moved to a different machine. For this purpose, the storage and network connectivity of the transferred VM or container also needs to be moved to the target physical machine [Cerroni and Callegati 2014]. In fog/edge computing, location-awareness should be considered for migration performance [Stojmenovic 2014].

INDICES [Shekhar et al. 2017] points out that server overloading needs to be addressed when a VM is migrated from a cloud data center to a fog cloud platform. INDICES considers the performance interference caused by resource contention between co-located VMs during VM migration. INDICES first identifies a user experiencing service level objective (SLO) violations and moves the user's VM to a fog cloud platform that can offer the lowest performance interference.

Bittencourt et al. [Bittencourt et al. 2015] detected the movement and behavior of a mobile device to decide where and when to migrate the user's VM among fog cloud platforms. When a user's device is disconnected from the access point of one fog cloud, the study identifies the user's location using a GPS system, and moves the user's VM to a nearby fog cloud. As data migration may incur a service suspension during migration, the research adopts a proactive technique that migrates the VM in advance, predicting the user's movement.

3.2.2. Network Virtualization. Network virtualization combines hardware and software network resources into a virtual network that is a software-based administrative entity for a tenant [Chowdhury and Boutaba 2010].

i. Software-Defined Networking (SDN) and Network Function Virtualization (NFV): A fog/edge cloud has an option to adopt software-defined networking (SDN) and network functions virtualization (NFV) for managing the network through software [Lee et al. 2018]. SDN separates the control plane from the data plane [Kreutz et al. 2015]. The control plane decides where the traffic is sent, and the data plane forwards the traffic to the destination decided by the control plane. NFV decouples networking functions such as routing and fire-walling from the underlying proprietary hardware, and allows each of the functions to run on a VM on commodity hardware [Han et al. 2015]. NFV is a complementary concept to SDN and is independent of it, although they are often combined together in modern clouds [Manzalini et al. 2013].

A virtual network enabled by SDN and NFV interconnects fog/edge clouds that are geographically dispersed [Bonomi et al. 2014]. The virtual network is required to support Layer 2 (L2) and Layer 3 (L3) networks, IPv4 and IPv6 protocols, and different addressing modes. Hybrid Fog and Cloud, called HFC [Moreno-Vozmediano et al. 2017], extends the BEACON [Moreno-Vozmediano et al. 2015] project that implements a federated cloud network for the efficient and automated provision of virtual networks to distributed fog/edge clouds. The framework installs an HFC agent in each cloud that

manages the control plane implemented by an SDN technology. The HFC agent also implements required Virtual Network Functions (VNFs) such as virtual switches and routers in order to interconnect the distributed clouds.

Constructing SDN and NFV in fog/edge platforms implies that clients can leverage elastic virtualized environments where all VMs for the same tenant can be in the same virtual LAN (VLAN) even if they are located in different areas. Wang et al. built an urban video surveillance system that exploits Virtualized Network Functions (VNFs) VMs as computational units for video analysis algorithms [Wang et al. 2017a]. More VMs can be allocated to higher priority tasks, and the source data can be sent between VMs by virtual switches controlled by the SDN routing strategies.

Conventional mobile clouds that offload tasks from mobile devices to centralized data centers are moving their applications to fog/edge clouds so as to reduce processing latency. NFV in fog/edge devices constructs a virtualized network infrastructure where computational resources can be scaled on the infrastructure based on demand. Yang et al. proposed a set of algorithms for dynamic resource allocation in such an NFV-enabled mobile fog/edge cloud [Yang et al. 2016; Yang et al. 2018]. An offline algorithm estimates the desired response time with minimum resources, and the auto-scaling and load-balancing algorithm makes provision for workload variations. When the capacity violation detection algorithm identifies a failure of the auto-scaling mechanism, a network latency constraint greedy algorithm initializes an NFV-enabled edge node to cope with the failure.

SDN is also applied to inter-vehicle communication using fifth generation (5G) vehicular networks or Vehicular Adhoc Network (VANET) [Vinel et al. 2017; Truong et al. 2015]. In this context, SDN can efficiently manage connected vehicles, called a vehicular neighbor group, with efficient member selection, group establishment, and flexible resource scheduling. 5G-SDVN abstracts vehicles on a 5G network as SDN switches and simplifies network management [Huang et al. 2017]. In this study, mobile fog computing is also exploited by considering vehicles as mobile users. As with 5G-SDVN, FSDN VANET applies both SDN and fog computing to connected vehicles on a VANET [Truong et al. 2015]. VANET is limited; it has long delays and unbalanced flow traffic when the number of vehicles increases. The separation of the control and data planes in SDN simplifies network management as the number of vehicles increases, while fog computing improves VANET services with additional computational capabilities.

In recent fog computing use cases, data tend to be internally generated and consumed between sensors [Ivanov et al. 2016]. In this setup, each fog node is expected to act as a wireless router to transfer data between sensors. Hakiri et al. [Hakiri et al. 2017] employed SDN for managing wireless fog networks. SDN generally adopts a centralized control plane, but the authors pointed out that this can be a single point of failure and might deteriorate reliability. This study developed a hybrid control plane in which a centralized controller manages the entire network, and additional controllers are attached during runtime to serve as backup should the centralized controller fail.

Huge traffic volumes from IoT devices can disrupt conventional IoT networks. SDN architectures can help to alleviate this problem. Xu et al. incorporated a Message Queuing Telemetry Transport (MQTT) that is an application layer protocol for IoT, with SDN-enabled fog computing [Xu et al. 2016; Karagiannis et al. 2015]. MQTT consists of publishers, subscribers, and the broker. The broker receives messages from a publisher and relays the published messages to subscribers. The study developed an SDN-based proxy broker where the broker acts as a control plane. The broker aggregates traffics from clients for effective transmission and utilizes an Open vSwitch (OVS) to forward traffic.

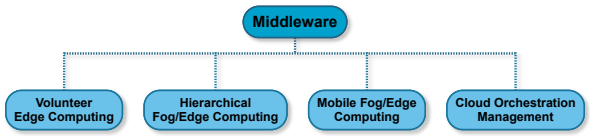

Fig. 14. A classification of middleware

ii. Overlay Network: An overlay network is a virtual network that is based on an underlying physical network and that provides additional network services (for example, peer-to-peer networks). Nodes in the overlay network are connected by virtual links to enable a new data path over physical links.

Koala [Tato et al. 2017] proposed an overlay network for decentralized edge computing. Different from cloud computing, there is no controller in decentralized edge computing, so each node has a limited view of the network. Koala built an overlay network to encourage collaboration between the decentralized nodes. However, proactively maintaining the overlay network incurs significant network traffic for identifying nodes joining or leaving the network. This is addressed by injecting maintenance messages into general applications traffic. Frugal [Aditya and Figueiredo 2017] focused on constructing an overlay network for online social networks. Frugal analyzed the social graphs between users and built a degree-constrained overlay topology using minimum degree-constrained spanning trees.

3.3. Middleware

Middleware provides complementary services to system software. Middleware in fog/edge computing provides performance monitoring, coordination and orchestration, communication facilities, protocols, and so on. Middleware used for fog/edge computing can be classified into four categories, as shown in Figure 14. The architecture of middleware is shown on the right side of Figure 13.

3.3.1. Volunteer Edge Computing. Nebula is middleware software that enables a decentralized edge cloud that consists of volunteer edge nodes that contribute resources [Chandra et al. 2013; Ryden et al. 2014]. Nebula comprises four major components. First, the Nebula Central provides a web-based portal for volunteer nodes and users to join the cloud and to deploy applications. Second, the DataStore, a data storage service, enables location-aware data processing. Third, the ComputePool offers computational resources to the volunteer edge nodes. Finally, the Nebula Monitor performs computation and network performance monitoring. In Nebula, the ComputePool coordinates with the DataStore to offer compute resources that have proximity to the input data in the DataStore.

Alonso-Monsalve et al. [Alonso-Monsalve et al. 2018] proposed a heterogeneous mobile cloud computing model where desktop computers or mobile devices donate their resources and form volunteer platforms. The authors pointed out that some clouds are experiencing network and computation saturation owing to a significant amount of user devices. To alleviate this situation, mobile users can exploit idle computation and storage resources of volunteer devices in geographically near places. As volunteer devices may not be long-lasting, the research also utilizes a cloud system that enables the applications to continue operations in the event of failures.

3.3.2. Hierarchical Fog/Edge Computing. A hierarchical fog/edge computing platform provides middleware that exploits both conventional cloud computing and recent fog/edge computing paradigms. Tasks that require prompt reaction are processed in fog/edge

nodes whereas complex or long-term analysis tasks are performed at more powerful cloud nodes [Hong et al. 2013; Tang et al. 2015; Nastic et al. 2017].

Mobile Fog facilitates hierarchical fog/edge computing. It enables easy communication between computing nodes at each hierarchical level and provides scaling capabilities during runtime [Hong et al. 2013]. In Mobile Fog, an application consists of three processes, each of which is mapped to a leaf node in smartphones or vehicles, an intermediate node in fog/edge computing, and a root node in the data center. Mobile Fog provides a range of APIs for communications and event handling between distributed processes. When a computing instance becomes congested, Mobile Fog creates a new instance at the same hierarchy level so as to load-balance workloads between nodes.

Tang et al. [Tang et al. 2015] proposed a hierarchical fog computing platform for processing big data generated by smart cities. The platform consists of four layers. The bottom layer, Layer 4, contains a massive number of sensor nodes that are widely distributed in public infrastructures. Layer 3 consists of low-power fog/edge devices that receive raw data from the sensor nodes in Layer 4. One fog/edge device is connected to nearby sensors to provide timely data analyses. Layer 2 comprises more powerful computing nodes, each of which is connected to a group of fog/edge devices in Layer 3. Layer 2 associates temporal data with spatial data to analyze potential risky events whereas Layer 3 focuses on immediate small threats. Layer 1 is a cloud computing platform that performs long-term analyses spanning a whole city by employing Hadoop.

Nastic et al. [Nastic et al. 2017] developed a unified edge and cloud platform for real-time data analytics. In this study, edge devices are used to execute simple data analytics such as measuring human vital signs sent from IoT mobile healthcare devices. Cloud computing receives preprocessed and filtered data from the edge devices and focuses on comprehensive data analytics to gain long-term insight about the person. The analytics function wrapper and API layer in the middleware provides a frontend for users to send and receive data to and from analytics functions in the cloud. The orchestration layer determines whether the provided data need to be processed in the edge or cloud node according to the high-level objectives of the application. Finally, the runtime mechanism layer schedules analytics functions and executes them while satisfying QoS requirements.

3.3.3. Mobile Fog/Edge Computing. Conventional mobile cloud computing [Chun et al. 2011; Kosta et al. 2012; López et al. 2016] allows low-power mobile devices such as smartphones to offload their computation-intensive tasks to more powerful platforms in cloud computing. This feature can improve the user experience and save power in mobile devices. However, cloud platforms in data centers cannot support low network latency and high bandwidth. To address this limitation, developers are exploiting fog/edge computing to offload their tasks for achieving satisfactory latency and bandwidth.

FemtoClouds [Habak et al. 2015] pay attention to recent powerful mobile devices such as smartphones and laptops, and form a compute cluster using these devices. A controller in FemtoClouds receives requests from users who installed the FemtoCloud service and schedules the requests in idle devices with sufficient capability. A business holder such as a coffee shop owner or a university can provide the controller. The Discovery Module in the controller discovers FemtoCloud devices and estimates the compute capacity of each one. Upon users' requests, the Execution Prediction Module predicts the completion time based on each device's execution load. The Task Assignment Module then iteratively assigns several tasks to less loaded devices to efficiently obtain the desired results.

Sensors Of Ubiquitous Life (SOUL) [Jang et al. 2016] constructs an edge-cloud for efficiently processing various sensors in mobile devices. The authors pointed out that

an application in a mobile device might not know how to handle device-specific sensors. SOUL provides APIs to virtualize sensors, thereby making it possible for diverse sensors to be treated in the same way. SOUL externalizes the virtualized sensors to the edge-cloud to leverage the cloud's computational and storage services. The SOUL Engine in each mobile device manages sensor-related operations executed by the application and sends these requests to the edge-cloud. The SOUL Core in the edge-cloud performs the received requests on behalf of the device. The two entities are connected by SOUL Streams.

Silva et al. [Silva et al. 2017] extended mobile cloud computing to an edge-cloud where nearby devices are connected by WiFi-Direct. The connected devices work together as a pool of computing resources for data caching and video streaming. Mobile devices that share the same interest (e.g., devices in the same sports stadium) establish a WiFi-Direct group. The cloud middleware tracks the members of the group along with their connection information and provides the content stored in each mobile device. This architecture relieves the load in the access points at a certain large venue and improves the quality of experience.

Human-driven edge computing (HEC) [Bellavista et al. 2018] points out that mobile edge computing has limitations because the number of edges is not sufficient, and some highly populated areas may result in congestion on edges. To address these limitations, HEC combines mobile edge computing with mobile crowdsensing [Ganti et al. 2011], where smartphones or tablet computers become edge nodes, share sensor data, and analyze the data for common interest. HEC does not implement a controller, but instead exploits local one-hop communications using VM/container migration between participants. The middleware for HEC consists of two components. *Elijah* is responsible for cloud resource management and migrates a VM to an identified edge node. The *Elijah* extension module additionally supports Docker-oriented containers and enables seamless VM/container migration when handoffs occur between different edge nodes.

3.3.4. Cloud Orchestration Management. In fog/edge computing, each device is regarded as a small compute server. The inter-device coordination for these devices is challenging compared to conventional clouds because (i) fog/edge devices have limited capabilities, (ii) the number of fog/edge devices expected to participate is greater than that of compute servers in a cloud data center, and (iii) fog/edge devices may be moving, and therefore the connectivity to the network may be intermittent [Amento et al. 2016].

While container-based cloud computing provides low overhead, a device in fog/edge computing still has resource limitations that cause the device to perform until it reaches the maximum computing capacity. The microCloud [Morabito and Beijar 2016] overcomes this limitation by exploiting resources of other edge devices. It adopts the Cloudy software [Selimi et al. 2015], an open source cloud management framework for local communities that is associated with Docker-based containers. Using the framework, a user can publish applications to a set of containers running on several edge devices. The microCloud thereby provides elasticity like other public clouds. The microCloud focused on local homogeneous devices, while Khan et al. [Khan and Freitag 2017] extended the concept of the microCloud to geographically distributed and heterogeneous devices.

Edge compute nodes may consist of thousands to millions of moving devices such as cars and drones. In this scenario, it is challenging to orchestrate the management of the devices using existing cloud management platforms such as OpenStack [Sefraoui et al. 2012]. FocusStack [Amento et al. 2016] introduces location-based awareness to OpenStack to deploy containers into devices that are geographically in the focus of attention. FocusStack minimizes managed devices at a single time by only paying attention to healthy devices in the target area. For this purpose, when a cloud op-

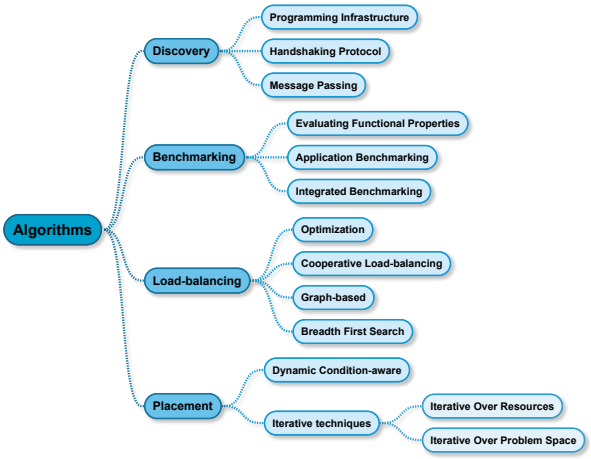

Fig. 15. A classification of algorithms in fog/edge computing

eration specifying certain requirements is invoked by a FocusStack API, the Geocast Georouter sends broadcast messages to edge devices in the target area to ask whether the devices can satisfy the request. If the Geocast Georouter receives responses from the devices, it regards them as healthy devices currently connected to the network. The Conductor component then sends the corresponding OpenStack operation to the selected devices. The minimization of managed devices at a single time allows FocusStack to be more efficient and scalable in edge clouds.

Foggy [Santoro et al. 2017] provided an orchestration tool for hierarchical fog computing that consists of the cloud (the highest tier), edge Cloudlets, edge gateways, and Swarm of Things tiers (the lowest tier near sensors). The Orchestrator deploys each Application Component, which is a module of a large application in a container image, on a node in each tier that satisfies user requirements.

Studies by Vogler and Nastic et al. [Vögler et al. 2015; Nastic et al. 2014; Nastic et al. 2016] introduced middleware for IoT clouds. In these studies, *software-defined IoT gateways (SDGs)* are defined for encapsulating infrastructure resources in a container. The IoT middleware focuses on the execution of provisioning workflows by supporting effective deployment of SDGs and customizing the SDGs to application-specific demands. When executing a provisioning workflow, the SDG manager decides compatible SDG images on a set of devices selected by the API manager. The Deployment Handler sends the selected SDG images to the Provisioning Daemon in each device that then starts the SGD and configures its virtual environment. Finally, the Provisioning Agent receives a specific application image from the Provisioning Daemon and installs and executes the image.

4. ALGORITHMS

There are several underlying algorithms used to facilitate fog/edge computing. In this section, we discuss four algorithms, namely (i) discovery - identifying edge resources within the network that can be used for distributed computation, (ii) benchmarking - capturing the performance of resources for decision-making to maximize the performance of deployments, (iii) load-balancing - distributing workloads across resources based on different criteria such as priorities, fairness, etc, and (iv) placement - identi-

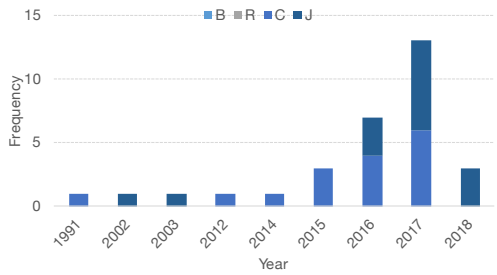

Fig. 16. A histogram of publications reviewed for the classification of the algorithms employed for resource management in fog/edge computing. Legend: B - books or book chapters; R - reports, including articles available on pre-print servers or white papers; C - conference or workshop papers; J - journal or magazine articles.

fying resources appropriate for deploying a workload. Figure 15 denotes this classification. A histogram of the research publications used is shown in Figure 16.

4.1. Discovery

Discovery refers to identifying edge resources so that workloads from the clouds or from user devices/sensors can be deployed on them. Typically, edge computing research assumes that edge resources are discovered. However, this is not an easy task [Varghese et al. 2016b]. Three techniques that use programming infrastructure, handshaking protocols, and message passing are employed in discovery.

The first technique uses *programming infrastructure* such as Foglets, proposed as a mechanism for edge resources to join a cloud-edge ecosystem [Saurez et al. 2016]. A discovery protocol was proposed that matches the resource requirements of an application against available resources on the edge. Nonetheless, the protocol assumes that the edge resource is publicly known or available for use. An additional join protocol is implemented that allows the selection of one edge node from among a set of resources that have the same geographic distance from the user.

The second technique uses *handshaking protocols*. The Edge-as-a-Service (EaaS) platform presents a lightweight discovery protocol for a collection of homogeneous edge resources [Varghese et al. 2017a]. The platform requires a master node that may be a compute available network device or a dedicated node that executes a manager process and communicates with edge nodes. The manager communicates with potential edge nodes and executes a process on the edge node to run commands. Once discovered, the Docker or LXD containers can be deployed on edge nodes.

The benefit of the EaaS platform is that the discovery protocol implemented is lightweight and the overhead is only a few seconds for launching, starting, stopping, or terminating containers. Up to 50 containers with an online game workload similar to PokeMon Go were launched on an individual edge node. However, this has been carried out in the context of a single collection of edge nodes. Further research will be required to employ such a model in a federated edge environment. The major drawback of the EaaS platform is that it assumes a centralized master node that can communicate with all potential edge nodes. The handshaking protocol assumes that the edge nodes can be queried and can be made available in a common marketplace via owners. In addition, the security-related implications of the master node installing a manager on the edge node and executing commands on it was not considered.

The third technique for discovery uses *message passing*. In the context of a sensor network in which the end devices may not necessarily have access to the Internet, there is research suggesting that messages may be delivered in such a network using

services offered by the nodes (referred to as processing nodes) connected to the Internet [Kolcun et al. 2015]. A discovery method for identifying the processing nodes was presented. The research assumed that a user can communicate with any node in a network and submit queries, and relies on simulation-based validation.

4.2. Benchmarking

Benchmarking is a de facto approach for capturing the performance (of entities such as memory, CPU, storage, network, etc) of a computing system [Varghese et al. 2014]. Metrics relevant to the performance of each entity need to be captured using standard performance evaluation tools. Typical tools used for clusters or supercomputers include LINPACK [Dongarra et al. 2003] or NAS Parallel Benchmarks [Bailey et al. 1991].

On the cloud, this is performed by running sample micro or macro applications that stress-tests each entity to obtain a snapshot of the performance of a Virtual Machine (VM) at any given point in time [Ferdman et al. 2012; Palit et al. 2016]. The key challenge of benchmarking a dynamic computing system (where workloads and conditions change significantly, such as the cloud and the edge) is obtaining metrics in near real-time [Varghese et al. 2014; 2016]. Existing benchmarking techniques for the cloud are time-consuming and are not practical solutions because they incur a lot of monetary costs. For example, accurately benchmarking a VM with 200 GB RAM and 1 TB storage requires a few hours. Alternate lightweight benchmarking techniques using containers have been proposed that can obtain results more quickly on the cloud than traditional techniques [Varghese et al. 2016a; Kozhirbayev and Sinnott 2017]. However, a few minutes are still required to get results comparable to traditional benchmarking.

Edge benchmarking can be classified into: (i) benchmarking for evaluating functional properties, (ii) application-based benchmarking, and (iii) integrated benchmarking. The majority of edge benchmarking research evaluates power, CPU, and memory performance of edge processors [Morabito 2017].

Benchmarking becomes more challenging in an edge environment for a number of reasons. First, because edge-specific *application benchmarks* that capture a variety of workloads are not yet available. Existing benchmarks are typically scientific applications that are less suited for the edge [Cherrueau et al. 2017]. Instead, voice-driven benchmarks [Sridhar and Tolentino 2017] and Internet-of-Things (IoT) applications have been used [Krylovskiy 2015]. Benchmarking object stores in edge environments have also been proposed [Confais et al. 2017b].

Second, running additional time-consuming applications on resource constrained edge nodes can be challenging. Spark has been evaluated in a highly resource constrained fog environment consisting of eight Raspberry Pi single-board computers [He et al. 2018]. The job completion time of Spark is reduced significantly with the cluster of Raspberry Pi computers, but it still requires a few minutes to get results. Instead of running time-consuming applications, CloudSim [Calheiros et al. 2011] has been used to simulate edge workloads for estimating resource usage and developing a pricing model in fog computing [Aazam and Huh 2015]. There is a need for lightweight benchmarking tools for the edge.

Finally, it is not sufficient to merely benchmark edge resources, but an *integrated approach for benchmarking cloud and edge resources* is required [Ficco et al. 2017]. This will ensure that the performance of all possible combinations of deployments of the application across the cloud and the edge is considered for maximizing overall application performance.

4.3. Load-Balancing

As edge data centers are deployed across the network edge, the issue of distributing tasks using an efficient load-balancing algorithm has gained significant attention.

Existing load-balancing algorithms at the edge employ four techniques, namely optimization techniques, cooperative load balancing, graph-based balancing, and using breadth-first search.

He et al. [He et al. 2016] proposed the Software Defined Cloud/Fog Networking (SDCFN) architecture for the Internet of Vehicles (IoV). SDCFN allows centralized control for networking between vehicles and helps the middleware to obtain the required information for load balancing. The study adopted *Particle Swarm Optimization - Constrained Optimization* (PSO-CO) [Parsopoulos et al. 2002] for load-balancing to decrease latency and effectively achieve the required quality of service (QoS) for vehicles.

CooLoad [Beraldi et al. 2017] proposed a *cooperative load-balancing* model between fog/edge data centers to decrease service suspension time. CooLoad assigns each data center a buffer to receive requests from clients. When the number of items in the buffer is above a certain threshold, incoming requests to the data center are load-balanced to an adjacent data center. This work assumed that the data centers were connected by a high-speed transport for effective load balancing.

Song et al. [Ningning et al. 2016] pointed out that existing load-balancing algorithms for cloud platforms that operate in a single cluster cannot be directly applied to a dynamic and peer-to-peer fog computing architecture. To realize efficient load-balancing, they abstracted the fog architecture as a *graph model* where each vertex indicates a node, and the graph edge denotes data dependency between tasks. A dynamic graph-repartitioning algorithm that uses previous load-balancing result as input and minimizes the difference between the load-balancing result and the original status was proposed.

Puthal et al. focused on developing an efficient dynamic load-balancing algorithm with an authentication method for edge data centers [Puthal et al. 2018]. Tasks were assigned to an under-utilized edge data center by applying the *Breadth First Search* (BFS) method. Each data center was modeled using the current load and the maximum capacity used to compute the current load. The authentication method allows the load-balancing algorithm to find an authenticated data center.

4.4. Placement

One challenging issue in fog/edge computing is to place incoming computation tasks on suitable fog/edge resources. Placement algorithms address this issue and need to consider the availability of resources in the fog/edge layer and the environmental changes [Dastjerdi et al. 2016]. Existing techniques can be classified as dynamic condition-aware techniques and iterative techniques. Iterative techniques can be further divided into two spaces: iterative over resources, and iterative over the problem spaces.

Wang et al. pointed out that existing work solved placement issues in fog/edge computing under static network conditions and predetermined resource demands and were not *dynamic condition-aware* (do not consider users' mobility and changes in resource availability) [Wang et al. 2017b]. This shortcoming was addressed by considering an additional set of parameters including the location and preference of a user, database location, and the load on the system. A method that predicts the values of the parameters when service instances of a user are placed in a certain configuration was proposed. The predicted values yielded an expected cost and optimal placement configuration with lowest cost. Ni et al. [Ni et al. 2017] predicted the completion time and price of a task based on priced timed Petri nets (PTPNs) in order to develop a resource allocation strategy to reduce latency and maximize resource utilization in fog computing. PTPN effectively deals with the dynamic behavior of the fog system to generate the performance and time cost [Abdulla and Mayr 2009].

Iterative methods over resources in the fog computing hierarchy is another effective technique. Taneja et al. [Taneja and Davy 2017] proposed a placement algorithm for hierarchical fog computing that exploits both conventional cloud and recent fog computing. The algorithm iterates from the fog towards the cloud for placing computation modules first on the available fog nodes. In this algorithm, a node is represented as a set of three attributes: the CPU, memory, and network bandwidth. Each computation module expresses its requirement in the form of the three attributes. The proposed solution first sorts the nodes and modules in ascending order to respectively associate the provided capacity with the requirement. The algorithm then places each module on an appropriate node that has enough resources, iterating from fog nodes to cloud nodes. The authors validated this algorithm using iFogSim, a fog computing simulation toolkit developed by Gupta et al. [Gupta et al. 2017].

Souza et al. [Souza et al. 2018] proposed service placement strategies for hierarchical Fog-to-Cloud (F2C) architectures in collaboration with service atomization and parallel execution. Service atomization divides a large service into smaller sub-services for workload distribution between the fog and the cloud. Parallel execution allows the divided sub-services to run on fog and cloud resources concurrently. Based on these techniques, the study suggests following three placement strategies: First-Fit (FF), Best-Fit (BF), and Best-fit with Queue (BQ). FF just selects available edge devices for allocation of sub-services, and if there are not available ones, FF sends the services to the cloud. BF sorts sub-services in ascending order based on the requested resources and allocates them to available edge devices. If the requested amount reaches a certain threshold for edge devices, the sub-services are sent to the cloud. BQ adopts BF as a basic strategy, but when edge devices are congested, it determines whether to send sub-services to the cloud or to queue them to the edge devices based on estimation.

In contrast to the above iterative method, *multiple iterations can be performed over the identified problem space*. Skarlat et al. proposed an approach called the Fog Service Placement Problem (FSPP) to optimally share resources in fog nodes among IoT services [Skarlat et al. 2017]. The FSPP considers QoS constraints such as latency or deadline requirements during placement. In the FSPP, a fog node is characterized by three attributes, the CPU, memory, and storage, similar to the work of Taneja et al. [Taneja and Davy 2017]. The FSPP suggests a proactive approach where the placement is performed periodically to meet the QoS requirement. When the response time of an application reaches the upper bound, the FSPP prioritizes the application and places it on a node that has enough resources. If there are not enough resources, the algorithm sends the service to the nearest fog network or cloud. The proposed model was evaluated on an extended iFogSim [Gupta et al. 2017].

5. CONCLUSIONS

In this survey, we noted that technical challenges to managing the limited resources in fog/edge computing have been addressed to a high degree. However, a few challenges still remain to be made to improve resource management in terms of the capabilities and performance of fog/edge computing. We discuss some future research directions to address the remaining challenges.

Fog/edge computing often employs resource-limited devices such as WiFi APs and set-top boxes that are not suitable for running heavyweight data processing tools such as Apache Spark and deep learning libraries. An alternative lightweight data processing tool such as Apache Quarks can be employed in resource-limited edge devices, but it lacks advanced data analytics functions. The imbalance between lightweight implementations and high performance needs to be addressed.

In fog/edge computing, containers are widely used because they realize lightweight virtualization. However, efficient accelerator management in containers has not been

explored sufficiently, compared to the research in virtual machines [Hong et al. 2017a; Montella et al. 2017]. In fog/edge devices, graphics processing units (GPUs), field programmable gate array (FPGAs), and tensor processing units (TPUs) can be employed for data analytics and deep learning algorithms [Varghese et al. 2018]. To reduce latency in the time-constrained workloads, accelerator scheduling algorithms that consider real-time characteristics are required in fog/edge computing.

Hierarchical fog/edge computing exploits both conventional cloud computing and recent fog/edge computing. In general, tasks that need prompt reaction are processed in fog/edge devices whereas long-term analysis tasks are performed at the cloud. However, it is challenging how to partition a single large workload into small tasks and distribute them to both the cloud and fog/edge for concurrent executions. An efficient partitioning method and task placement strategy based on accurate prediction are required.

There are only a few infrastructure options available for pursuing real fog/edge computing environments. Many academic researchers rely on simulation studies using tools, such as iFogSim [Gupta et al. 2017] and EdgeCloudSim [Sonmez et al. 2018]. Miniature experimental environments have been also set up. For example, using single-board computers (SBCs) such as Odroid [Wang et al. 2017c] or Raspberry Pi boards [Johnston et al. 2018]. More realistic and large-scale testbeds have been recently set up, but have limited public availability. These include fog/edge testbeds implemented by Raytheon BBN Technologies [Gosain et al. 2016], the Institut National de la Recherche Scientifique [Muñoz et al. 2017], Optical Networks and Systems Department [Rimal et al. 2018], and Princeton University [Consortium 2017]. An effort to build realistic and large-scale fog/edge testbeds is required.

Fog/edge computing has gained significant attention over the last few years as an alternative approach to the conventional centralized cloud computing model. It brings computing resources close to mobile and IoT devices to reduce communication latency and enable efficient use of the network bandwidth. In this survey paper, research on resource management techniques in fog/edge computing was studied to identify and classify the key contributions in the three areas of architectures, infrastructure, and algorithms.

ACKNOWLEDGMENTS

The authors are grateful to the anonymous reviewers for their valuable comments and suggestions.

REFERENCES

Mohammad Aazam and Eui-Nam Huh. 2014. Fog Computing and Smart Gateway Based Communication for Cloud of Things. In *2014 International Conference on Future Internet of Things and Cloud (FiCloud)*. IEEE, 464–470.

Mohammad Aazam and Eui-Nam Huh. 2015. Fog computing micro datacenter based dynamic resource estimation and pricing model for IoT. In *2015 IEEE 29th International Conference on Advanced Information Networking and Applications*. IEEE, 687–694.

Parosh Aziz Abdulla and Richard Mayr. 2009. Minimal cost reachability/coverability in priced timed Petri nets. In *International Conference on Foundations of Software Science and Computational Structures*. Springer, 348–363.

Saumitra Aditya and Renato J Figueiredo. 2017. Frugal: Building Degree-Constrained Overlay Topology from Social Graphs. In *2017 IEEE 1st International Conference on Fog and Edge Computing (ICFEC)*. IEEE, 11–20.

Raja Wasim Ahmad, Abdullah Gani, Siti Hafizah Ab Hamid, Muhammad Shiraz, Abdullah Yousafzai, and Feng Xia. 2015. A Survey on Virtual Machine Migration and Server Consolidation Frameworks for Cloud Data Centers. *Journal of Network and Computer Applications* 52 (2015), 11–25.

Saúl Alonso-Monsalve, Félix García-Carballeira, and Alejandro Calderón. 2018. A heterogeneous mobile cloud computing model for hybrid clouds. *Future Generation Computer Systems* 87 (2018), 651–666.

Brian Amento, Bharath Balasubramanian, Robert J Hall, Kaustubh Joshi, Gueyoung Jung, and K Hal Purdy. 2016. FocusStack: Orchestrating Edge Clouds Using Location-Based Focus of Attention. In *IEEE/ACM Symposium on Edge Computing (SEC)*. IEEE, 179–191.

Christos Anagnostopoulos. 2014. Time-optimized contextual information forwarding in mobile sensor networks. *J. Parallel and Distrib. Comput.* 74, 5 (2014), 2317–2332.

Miguel Báguena, George Samaras, Andreas Pamboris, Mihail L Sichitiu, Peter Pietzuch, and Pietro Manzoni. 2016. Towards enabling hyper-responsive mobile apps through network edge assistance. In *2016 13th IEEE Annual Consumer Communications & Networking Conference (CCNC)*. IEEE, 399–404.

David H Bailey, Eric Barszcz, John T Barton, David S Browning, Robert L Carter, Leonardo Dagum, Rod A Fatoohi, Paul O Frederickson, Thomas A Lasinski, Rob S Schreiber, and others. 1991. The NAS parallel benchmarks summary and preliminary results. In *Supercomputing'91: Proceedings of the 1991 ACM/IEEE conference on Supercomputing*. IEEE, 158–165.

Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. 2003. Xen and the Art of Virtualization. In *ACM SIGOPS operating systems review*, Vol. 37. 164–177.

Luca Becchetti, Alberto Marchetti-Spaccamela, Andrea Vitaletti, Peter Korteweg, Martin Skutella, and Leen Stougie. 2009. Latency-constrained aggregation in sensor networks. *ACM Transactions on Algorithms (TALG)* 6, 1 (2009), 13.

Paolo Bellavista, Stefano Chessa, Luca Foschini, Leo Gioia, and Michele Girolami. 2018. Human-Enabled Edge Computing: Exploiting the Crowd as a Dynamic Extension of Mobile Edge Computing. *IEEE Communications Magazine* 56, 1 (2018), 145–155.

Paolo Bellavista and Alessandro Zanni. 2017. Feasibility of Fog Computing Deployment based on Docker Containerization over Raspberry Pi. In *Proceedings of the 18th International Conference on Distributed Computing and Networking*. ACM, 16.

Roberto Beraldi, Abderrahmen Mtibaa, and Hussein Alnuweiri. 2017. Cooperative load balancing scheme for edge computing resources. In *2017 Second International Conference on Fog and Mobile Edge Computing (FMEC)*. IEEE, 94–100.

Ketan Bhardwaj, Pragya Agrawal, Ada Gavrilovska, and Karsten Schwan. 2015. Appsachet: Distributed app delivery from the edge cloud. In *International Conference on Mobile Computing, Applications, and Services*. Springer, 89–106.

Arani Bhattcharya and Pradipta De. 2016. Computation offloading from mobile devices: Can edge devices perform better than the cloud?. In *Proceedings of the Third International Workshop on Adaptive Resource Management and Scheduling for Cloud Computing*. ACM, 1–6.

Luiz Fernando Bittencourt, Márcio Moraes Lopes, Ioan Petri, and Omer F Rana. 2015. Towards Virtual Machine Migration in Fog Computing. In *10th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing*. IEEE, 1–8.

Flavio Bonomi, Rodolfo Milito, Preethi Natarajan, and Jiang Zhu. 2014. Fog Computing: A Platform for Internet of Things and Analytics. In *Big data and internet of things: A roadmap for smart environments*. 169–186.

Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. 2012. Fog computing and its role in the internet of things. In *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*. ACM, 13–16.

Eleonora Borgia, Raffaele Bruno, Marco Conti, Davide Mascitti, and Andrea Passarella. 2016. Mobile edge clouds for Information-Centric IoT services. In *2016 IEEE symposium on computers and communication (ISCC)*. IEEE, 422–428.

Eric A Brewer. 2015. Kubernetes and the Path to Cloud Native. In *Proceedings of the Sixth ACM Symposium on Cloud Computing*. 167–167.

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

Rodrigo N Calheiros, Rajiv Ranjan, Anton Beloglazov, César AF De Rose, and Rajkumar Buyya. 2011. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and experience* 41, 1 (2011), 23–50.

Guilhem Castagnos and Fabien Laguillaumie. 2015. Linearly homomorphic encryption from DDH. In *Cryptographers' Track at the RSA Conference*. Springer, 487–505.

Walter Cerroni and Franco Callegati. 2014. Live Migration of Virtual Network Functions in Cloud-based Edge Networks. In *IEEE International Conference on Communications*. IEEE, 2963–2968.

Abhishek Chandra, Jon Weissman, and Benjamin Heintz. 2013. Decentralized Edge Clouds. *IEEE Internet Computing* 17, 5 (2013), 70–73.

Long Chen, Jigang Wu, Xin Long, and Zikai Zhang. 2017. ENGINE: Cost Effective Offloading in Mobile Edge Computing with Fog-Cloud Cooperation. (2017). arXiv:1711.01683. Retrieved from https://arxiv.org/abs/1711.01683.

Xu Chen. 2015. Decentralized computation offloading game for mobile cloud computing. *IEEE Transactions on Parallel and Distributed Systems* 26, 4 (2015), 974–983.

Xu Chen, Lei Jiao, Wenzhong Li, and Xiaoming Fu. 2016. Efficient multi-user computation offloading for mobile-edge cloud computing. *IEEE/ACM Transactions on Networking* 24, 5 (2016), 2795–2808.

Xu Chen and Junshan Zhang. 2017. When D2D meets cloud: Hybrid mobile task offloadings in fog computing. In *2017 IEEE international conference on communications (ICC)*. IEEE, 1–6.

Yanan Chen, Zhenyu Lu, Hu Xiong, and Weixiang Xu. 2018. Privacy-Preserving Data Aggregation Protocol for Fog Computing-Assisted Vehicle-to-Infrastructure Scenario. *Security and Communication Networks* (2018), 14.

Zhuo Chen, Lu Jiang, Wenlu Hu, Kiryong Ha, Brandon Amos, Padmanabhan Pillai, Alex Hauptmann, and Mahadev Satyanarayanan. 2015. Early implementation experience with wearable cognitive assistance applications. In *Proceedings of the 2015 workshop on Wearable Systems and Applications*. ACM, 33–38.

Ronan-Alexandre Cherrueau, Dimitri Pertin, Anthony Simonet, Adrien Lebre, and Matthieu Simonin. 2017. Toward a holistic framework for conducting scientific evaluations of openstack. In *Proceedings of the 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*. IEEE Press, 544–548.

Susanta Nanda Tzi-cker Chiueh and Stony Brook. 2005. A survey on virtualization technologies. *Rpe Report* 142 (2005).

NM Mosharaf Kabir Chowdhury and Raouf Boutaba. 2010. A survey of network virtualization. *Computer Networks* 54, 5 (2010), 862–876.

Byung-Gon Chun, Sunghwan Ihm, Petros Maniatis, Mayur Naik, and Ashwin Patti. 2011. Clonecloud: elastic execution between mobile device and cloud. In *Proceedings of the sixth conference on Computer systems*. ACM, 301–314.

Bastien Confais, Adrien Lebre, and Benoît Parrein. 2017a. An Object Store Service for a Fog/Edge Computing Infrastructure Based on IPFS and a Scale-Out NAS. In *IEEE 1st International Conference on Fog and Edge Computing*. 41–50.

Bastien Confais, Adrien Lebre, and Benoît Parrein. 2017b. Performance analysis of object store systems in a fog and edge computing infrastructure. In *Transactions on Large-Scale Data-and Knowledge-Centered Systems XXXIII*. Springer, 40–79.

OpenFog Consortium. 2017. OpenFog Reference Architecture for Fog Computing. (2017). Retrieved March 8, 2018 from https://www.openfogconsortium.org/wp-content/uploads/OpenFog_Reference_Architecture_2_09_17-FINAL.pdf.

Yong Cui, Jian Song, Kui Ren, Minming Li, Zongpeng Li, Qingmei Ren, and Yangjun Zhang. 2017. Software Defined Cooperative Offloading for Mobile Cloudlets. *IEEE/ACM Transactions on Networking* 25, 3 (2017), 1746–1760.

Amir Vahid Dastjerdi and Rajkumar Buyya. 2016. Fog computing: Helping the Internet of Things realize its potential. *Computer* 49, 8 (2016), 112–116.

Amir Vahid Dastjerdi, Harshit Gupta, Rodrigo N Calheiros, Soumya K Ghosh, and Rajkumar Buyya. 2016. Fog computing: Principles, architectures, and applications. In *Internet of Things*. Elsevier, 61–75.

Marcos Dias de Assunção, Alexandre da Silva Veith, and Rajkumar Buyya. 2018. Distributed Data Stream Processing and Edge Computing: A Survey on Resource Elasticity and Future Directions. *Journal of Network and Computer Applications* 103 (2018), 1 – 17.

Hoang T. Dinh, Chonho Lee, Dusit Niyato, and Ping Wang. 2013. A Survey of Mobile Cloud Computing: Architecture, Applications, and Approaches. *Wireless Communications and Mobile Computing* 13, 18 (2013), 1587–1611.

Jack J Dongarra, Piotr Luszczek, and Antoine Petitet. 2003. The LINPACK benchmark: past, present and future. *Concurrency and Computation: practice and experience* 15, 9 (2003), 803–820.

Ivan Farris, Leonardo Militano, Michele Nitti, Luigi Atzori, and Antonio Iera. 2017. MIFaaS: A mobile-IoT-federation-as-a-service model for dynamic cooperation of IoT cloud providers. *Future Generation Computer Systems* 70 (2017), 126–137.

Elena Fasolo, Michele Rossi, Jorg Widmer, and Michele Zorzi. 2007. In-network aggregation techniques for wireless sensor networks: a survey. *IEEE Wireless Communications* 14, 2 (2007), 70–87.

Michael Ferdman, Almutaz Adileh, Onur Kocberber, Stavros Volos, Mohammad Alisafaee, Djordje Jevdjic, Cansu Kaynak, Adrian Daniel Popescu, Anastasia Ailamaki, and Babak Falsafi. 2012. Clearing the clouds: a study of emerging scale-out workloads on modern hardware. 47, 4 (2012), 37–48.

Niroshinie Fernando, Seng W Loke, and Wenny Rahayu. 2016. Computing with nearby mobile devices: a work sharing algorithm for mobile edge-clouds. *IEEE Transactions on Cloud Computing* (2016).

Massimo Ficco, Christian Esposito, Yang Xiang, and Francesco Palmieri. 2017. Pseudo-dynamic testing of realistic edge-fog cloud ecosystems. *IEEE Communications Magazine* 55, 11 (2017), 98–104.

Mattias Forsman, Andreas Glad, Lars Lundberg, and Dragos Ilie. 2015. Algorithms for Automated Live Migration of Virtual Machines. *Journal of Systems and Software* 101 (2015), 110–126.

Colin Funai, Cristiano Tapparello, and Wendi Heinzelman. 2016. Mobile to mobile computational offloading in multi-hop cooperative networks. In *2016 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 1–7.

Raghu K Ganti, Fan Ye, and Hui Lei. 2011. Mobile Crowdsensing: Current State and Future Challenges. *IEEE Communications Magazine* 49, 11 (2011).

Lei Gao, Mike Dahlin, Amol Nayate, Jiandan Zheng, and Arun Iyengar. 2003. Application specific data replication for edge services. In *Proceedings of the 12th international conference on World Wide Web*. ACM, 449–460.

Wei Gao. 2014. Opportunistic peer-to-peer mobile cloud computing at the tactical edge. In *2014 IEEE Military Communications Conference*. IEEE, 1614–1620.

Nam Ky Giang, Michael Blackstock, Rodger Lea, and Victor CM Leung. 2015. Developing iot applications in the fog: A distributed dataflow approach. In *2015 5th International Conference on the Internet of Things (IOT)*. IEEE, 155–162.

Abhimanyu Gosain, Mark Berman, Marshall Brinn, Thomas Mitchell, Chuan Li, Yuehua Wang, Hai Jin, Jing Hua, and Hongwei Zhang. 2016. Enabling Campus Edge Computing Using Geni Racks and Mobile Resources. In *IEEE/ACM Symposium on Edge Computing (SEC)*. 41–50.

Lin Gu, Deze Zeng, Song Guo, Ahmed Barnawi, and Yong Xiang. 2017. Cost Efficient Resource Management in Fog Computing Supported Medical Cyber-Physical System. *IEEE Transactions on Emerging Topics in Computing* 5, 1 (2017), 108–119.

Harshit Gupta, Amir Vahid Dastjerdi, Soumya K Ghosh, and Rajkumar Buyya. 2017. iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments. *Software: Practice and Experience* 47, 9 (2017), 1275–1296.

Karim Habak, Mostafa Ammar, Khaled A. Harras, and Ellen Zegura. 2015. Femto Clouds: Leveraging Mobile Devices to Provide Cloud Service at the Edge. In *Proceedings of the IEEE 8th International Conference on Cloud Computing*. 9–16.

Karim Habak, Ellen W Zegura, Mostafa Ammar, and Khaled A Harras. 2017. Workload management for dynamic mobile device clusters in edge femtoclouds. In *Proceedings of the Second ACM/IEEE Symposium on Edge Computing*. ACM, 6.

Muhammad Habib ur Rehman, Prem Prakash Jayaraman, Saif ur Rehman Malik, Atta ur Rehman Khan, and Mohamed Medhat Gaber. 2017. RedEdge: A Novel Architecture for Big Data Processing in Mobile Edge Computing Environments. *Journal of Sensor and Actuator Networks* 6, 3 (2017).

Akram Hakiri, Bassem Sellami, Prithviraj Patil, Pascal Berthou, and Aniruddha Gokhale. 2017. Managing Wireless Fog Networks using Software-Defined Networking. In *IEEE/ACS 14th International Conference on Computer Systems and Applications*.

Bo Han, Vijay Gopalakrishnan, Lusheng Ji, and Seungjoon Lee. 2015. Network Function Virtualization: Challenges and Opportunities for Innovations. *IEEE Communications Magazine* 53, 2 (2015), 90–97.

Natascha Harth and Christos Anagnostopoulos. 2017. Quality-aware aggregation & predictive analytics at the edge. In *2017 IEEE International Conference on Big Data (Big Data)*. IEEE, 17–26.

Nicholas Haydel, Sandra Gesing, Ian Taylor, Gregory Madey, Abdul Dakkak, Simon Garcia De Gonzalo, and Wen-Mei W Hwu. 2015. Enhancing the Usability and Utilization of Accelerated Architectures via Docker. In *IEEE/ACM 8th International Conference on Utility and Cloud Computing*. 361–367.

Jing He, Shouling Ji, Yi Pan, and Yingshu Li. 2014. Constructing load-balanced data aggregation trees in probabilistic wireless sensor networks. *IEEE Transactions on Parallel and Distributed Systems* 25, 7 (2014), 1681–1690.

Jianhua He, Jian Wei, Kai Chen, Zuoyin Tang, Yi Zhou, and Yan Zhang. 2018. Multitier fog computing with large-scale iot data analytics for smart cities. *IEEE Internet of Things Journal* 5, 2 (2018), 677–686.

Xiuli He, Zhiyuan Ren, Chenhua Shi, and Jian Fang. 2016. A Novel Load Balancing Strategy of Software-defined Cloud/Fog Networking in the Internet of Vehicles. *China Communications* 13, 2 (2016), 140–149.

Matt Helsley. 2009. LXC: Linux Container Tools. *IBM developerWorks Technical Library* 11 (2009).

Cheol-Ho Hong, Kyungwoon Lee, Minkoo Kang, and Chuck Yoo. 2018. qCon: QoS-Aware Network Resource Management for Fog Computing. *Sensors* 18, 10 (2018), 3444.

Cheol-Ho Hong, Ivor Spence, and Dimitrios S Nikolopoulos. 2017a. FairGV: fair and fast GPU virtualization. *IEEE Transactions on Parallel and Distributed Systems* 28, 12 (2017), 3472–3485.

Cheol-Ho Hong, Ivor Spence, and Dimitrios S Nikolopoulos. 2017b. GPU Virtualization and Scheduling Methods: A Comprehensive Survey. *ACM Computing Surveys (CSUR)* 50, 3 (2017), 35.

Hua-Jun Hong. 2017. From Cloud Computing to Fog Computing: Unleash the Power of Edge and End Devices. In *IEEE International Conference on Cloud Computing Technology and Science*. IEEE, 331–334.

Hua-Jun Hong, Jo-Chi Chuang, and Cheng-Hsin Hsu. 2016. Animation Rendering on Multimedia Fog Computing Platforms. In *IEEE International Conference on Cloud Computing Technology and Science*. IEEE, 336–343.

Kirak Hong, David Lillethun, Umakishore Ramachandran, Beate Ottenwälder, and Boris Koldehofe. 2013. Mobile Fog: A Programming Model for Large-scale Applications on the Internet of Things. In *Proceedings of the second ACM SIGCOMM workshop on Mobile cloud computing*. 15–20.

Xumin Huang, Rong Yu, Jiawen Kang, Yejun He, and Yan Zhang. 2017. Exploring Mobile Edge Computing for 5G-enabled Software Defined Vehicular Networks. *IEEE Wireless Communications* 24, 6 (2017), 55–63.

Stepan Ivanov, Sasitharan Balasubramaniam, Dmitri Botvich, and Ozgur B Akan. 2016. Gravity Gradient Routing for Information Delivery in Fog Wireless Sensor Networks. *Ad Hoc Networks* 46 (2016), 61–74.

Minsung Jang, Hyunjong Lee, Karsten Schwan, and Ketan Bhardwaj. 2016. SOUL: An Edge-Cloud System for Mobile Applications in a Sensor-rich World. In *IEEE/ACM Symposium on Edge Computing*. 155–167.

Vimalkumar Jeyakumar, Mohammad Alizadeh, David Mazières, Balaji Prabhakar, Albert Greenberg, and Changhoon Kim. 2013. EyeQ: Practical network performance isolation at the edge. In *Presented as part of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)*. 297–311.

Hongbo Jiang, Shudong Jin, and Chonggang Wang. 2011. Prediction or not? An energy-efficient framework for clustering-based data collection in wireless sensor networks. *IEEE Transactions on Parallel and Distributed Systems* 22, 6 (2011), 1064–1071.

Steven J. Johnston, Philip J. Basford, Colin S. Perkins, Herry Herry, Fung Po Tso, Dimitrios Pezaros, Robert D. Mullins, Eiko Yoneki, Simon J. Cox, and Jeremy Singer. 2018. Commodity Single Board Computer Clusters and Their Applications. *Future Generation Computer Systems* (June 2018).

Yiping Kang, Johann Hauswald, Cao Gao, Austin Rovinski, Trevor Mudge, Jason Mars, and Lingjia Tang. 2017. Neurosurgeon: Collaborative intelligence between the cloud and mobile edge. *Acm Sigplan Notices* 52, 4 (2017), 615–629.

Vasileios Karagiannis, Periklis Chatzimisios, Francisco Vazquez-Gallego, and Jesus Alonso-Zarate. 2015. A Survey on Application Layer Protocols for the Internet of Things. *Transaction on IoT and Cloud Computing* 3, 1 (2015), 11–17.

Kuljeet Kaur, Tanya Dhand, Neeraj Kumar, and Sherali Zeadally. 2017. Container-as-a-service at the edge: Trade-off between energy efficiency and service availability at fog nano data centers. *IEEE wireless communications* 24, 3 (2017), 48–56.

Yasaman Keshtkarjahromi, Yuxuan Xing, and Hulya Seferoglu. 2018. Dynamic heterogeneity-aware coded cooperative computation at the edge. In *2018 IEEE 26th International Conference on Network Protocols (ICNP)*. IEEE, 23–33.

Amin M Khan and Felix Freitag. 2017. On Edge Cloud Service Provision with Distributed Home Servers. In *2017 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*. IEEE, 223–226.

Dragi Kimovski, Humaira Ijaz, Nishant Saurabh, and Radu Prodan. 2018. Adaptive nature-inspired fog architecture. In *2018 IEEE 2nd International Conference on Fog and Edge Computing (ICFEC)*. IEEE, 1–8.

Avi Kivity, Yaniv Kamay, Dor Laor, Uri Lublin, and Anthony Liguori. 2007. KVM: The Linux Virtual Machine Monitor. In *Proceedings of the Linux symposium*, Vol. 1. 225–230.

Roman Kolcun, David Boyle, and Julie A McCann. 2015. Optimal processing node discovery algorithm for distributed computing in IoT. In *2015 5th International Conference on the Internet of Things (IOT)*. IEEE, 72–79.

Sokol Kosta, Andrius Aucinas, Pan Hui, Richard Mortier, and Xinwen Zhang. 2012. Thinkair: Dynamic Resource Allocation and Parallel Execution in the Cloud for Mobile Code Offloading. In *Infocom, 2012 Proceedings IEEE*. 945–953.

Zhanibek Kozhirbayev and Richard O. Sinnott. 2017. A Performance Comparison of Container-based Technologies for the Cloud. *Future Generation Computer Systems* 68 (2017), 175 – 182.

Diego Kreutz, Fernando MV Ramos, Paulo Esteves Verissimo, Christian Esteve Rothenberg, Siamak Azodolmolky, and Steve Uhlig. 2015. Software-Defined Networking: A Comprehensive Survey. *Proc. IEEE* 103, 1 (2015), 14–76.

Alexandr Krylovskiy. 2015. Internet of things gateways meet linux containers: Performance evaluation and discussion. In *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*. IEEE, 222–227.

Insup Lee, Oleg Sokolsky, Sanjian Chen, John Hatcliff, Eunkyoung Jee, BaekGyu Kim, Andrew King, Margaret Mullen-Fortino, Soojin Park, Alexander Roederer, and others. 2012. Challenges and Research Directions in Medical Cyber–Physical Systems. *Proc. IEEE* 100, 1 (2012), 75–90.

Kyungwoon Lee, Chiyoung Lee, Cheol-Ho Hong, and Chuck Yoo. 2018. Enhancing the Isolation and Performance of Control Planes for Fog Computing. *Sensors* 18, 10 (2018), 3267.

Hongxing Li, Chuan Wu, Qiang-Sheng Hua, and Francis C. M. Lau. 2014. Latency-minimizing Data Aggregation in Wireless Sensor Networks Under Physical Interference Model. *Ad Hoc Networks* 12 (Jan. 2014), 52–68.

Hongxing Li, Chuan Wu, Dongxiao Yu, Qiang-Sheng Hua, and Francis CM Lau. 2013. Aggregation latency-energy tradeoff in wireless sensor networks with successive interference cancellation. *IEEE Transactions on Parallel and Distributed Systems* 24, 11 (2013), 2160–2170.

Yi Lin, Bettina Kemme, Marta Patino-Martinez, and Ricardo Jimenez-Peris. 2007. Enhancing edge computing with database replication. In *2007 26th IEEE International Symposium on Reliable Distributed Systems (SRDS 2007)*. IEEE, 45–54.

Peng Liu, Dale Willis, and Suman Banerjee. 2016. Paradrop: Enabling lightweight multi-tenancy at the network's extreme edge. In *2016 IEEE/ACM Symposium on Edge Computing (SEC)*. IEEE, 1–13.

Lara López, Francisco Javier Nieto, Terpsichori-Helen Velivassaki, Sokol Kosta, Cheol-Ho Hong, Raffaele Montella, Iakovos Mavroidis, and Carles Fernández. 2016. Heterogeneous secure multi-level remote acceleration service for low-power integrated systems and devices. *Procedia Computer Science* 97 (2016), 118–121.

Rongxing Lu, Kevin Heung, Arash Habibi Lashkari, and Ali A Ghorbani. 2017. A lightweight privacy-preserving data aggregation scheme for fog computing-enhanced IoT. *IEEE Access* 5 (2017), 3302–3312.

Xiao Ma, Chuang Lin, Xudong Xiang, and Congjie Chen. 2015. Game-theoretic analysis of computation offloading for cloudlet-based mobile cloud computing. In *Proceedings of the 18th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*. ACM, 271–278.

Redowan Mahmud, Satish Narayana Srirama, Kotagiri Ramamohanarao, and Rajkumar Buyya. 2018. Quality of Experience (QoE)-aware Placement of Applications in Fog Computing Environments. *J. Parallel and Distrib. Comput.* (2018).

Amit Manjhi, Suman Nath, and Phillip B. Gibbons. 2005. Tributaries and Deltas: Efficient and Robust Aggregation in Sensor Network Streams. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 287–298.

Dnyaneshwar Mantri, Neeli Rashmi Prasad, and Ramjee Prasad. 2013. BHCDA: Bandwidth efficient heterogeneity aware cluster based data aggregation for Wireless Sensor Network. In *2013 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. IEEE, 1064–1069.

Dnyaneshwar S. Mantri, Neeli Rashmi Prasad, and Ramjee Prasad. 2015. Bandwidth Efficient Cluster-based Data Aggregation for Wireless Sensor Network. *Computers and Electrical Engineering* 41, C (Jan. 2015), 256–264.

Dnyaneshwar S. Mantri, Neeli Rashmi Prasad, and Ramjee Prasad. 2016. Mobility and Heterogeneity Aware Cluster-Based Data Aggregation for Wireless Sensor Network. *Wireless Personal Communications* 86, 2 (01 Jan 2016), 975–993.

Antonio Manzalini, Roberto Minerva, Franco Callegati, Walter Cerroni, and Aldo Campi. 2013. Clouds of Virtual Machines in Edge Networks. *IEEE Communications Magazine* 51, 7 (2013), 63–70.

Yuyi Mao, Changsheng You, Jun Zhang, Kaibin Huang, and Khaled B Letaief. 2017. A survey on mobile edge computing: The communication perspective. *IEEE Communications Surveys & Tutorials* 19, 4 (2017), 2322–2358.

Violeta Medina and Juan Manuel García. 2014. A survey of migration mechanisms of virtual machines. *ACM Computing Surveys (CSUR)* 46, 3 (2014), 30.

Dirk Merkel. 2014. Docker: Lightweight Linux Containers for Consistent Development and Deployment. *Linux Journal* 2014, 239 (2014), 2.

Dejan S Milojicic, Vana Kalogeraki, Rajan Lukose, Kiran Nagaraja, Jim Pruyne, Bruno Richard, Sami Rollins, and Zhichen Xu. 2002. *Peer-to-peer computing*. Technical Report.

Nitinder Mohan and Jussi Kangasharju. 2016. Edge-Fog cloud: A distributed cloud for Internet of Things computations. In *2016 Cloudification of the Internet of Things (CIoT)*. IEEE, 1–6.

Raffaele Montella, Giulio Giunta, Giuliano Laccetti, Marco Lapegna, Carlo Palmieri, Carmine Ferraro, Valentina Pelliccia, Cheol-Ho Hong, Ivor Spence, and Dimitrios S Nikolopoulos. 2017. On the virtualization of CUDA based GPU remoting on ARM and X86 machines in the GVirtuS framework. *International Journal of Parallel Programming* 45, 5 (2017), 1142–1163.

Roberto Morabito. 2017. Virtualization on internet of things edge devices with container technologies: a performance evaluation. *IEEE Access* 5 (2017), 8835–8850.

Roberto Morabito and Nicklas Beijar. 2016. Enabling Data Processing at the Network Edge Through Lightweight Virtualization Technologies. In *IEEE International Conference on Sensing, Communication and Networking*. 1–6.

Roberto Morabito, Vittorio Cozzolino, Aaron Yi Ding, Nicklas Beijar, and Jorg Ott. 2018. Consolidate IoT edge computing with lightweight virtualization. *IEEE Network* 32, 1 (2018), 102–111.

Rafael Moreno-Vozmediano, Eduardo Huedo, Ignacio M Llorente, Rubén S Montero, Philippe Massonet, Massimo Villari, Giovanni Merlino, Antonio Celesti, Anna Levin, Liran Schour, and others. 2015. BEACON: A Cloud Network Federation Framework. In *European Conference on Service-Oriented and Cloud Computing*. 325–337.

Rafael Moreno-Vozmediano, Ruben S Montero, Eduardo Huedo, and Ignacio M Llorente. 2017. Cross-site Virtual Network in Cloud and Fog Computing. *IEEE Cloud Computing* 4, 2 (2017), 46–53.

Abderrahmen Mtibaa, Afnan Fahim, Khaled A. Harras, and Mostafa H. Ammar. 2013. Towards Resource Sharing in Mobile Device Clouds: Power Balancing Across Mobile Devices. *ACM SIGCOMM Computer Communication Review* 43, 4 (Aug. 2013), 51–56.

Abderrahmen Mtibaa, Khaled Harras, and Hussein Alnuweiri. 2015. Friend or foe? Detecting and isolating malicious nodes in mobile edge computing platforms. In *2015 IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom)*. IEEE, 42–49.

Raul Muñoz, Laia Nadal, Ramon Casellas, Michela Svaluto Moreolo, Ricard Vilalta, Josep Maria Fàbrega, Ricardo Martínez, Arturo Mayoral, and Fco Javier Vílchez. 2017. The ADRENALINE testbed: An SDN/NFV packet/optical transport network and edge/core cloud platform for end-to-end 5G and IoT services. In *2017 European Conference on Networks and Communications (EuCNC)*. IEEE, 1–5.

Stefan Nastic, Thomas Rausch, Ognjen Scekic, Schahram Dustdar, Marjan Gusev, Bojana Koteska, Magdalena Kostoska, Boro Jakimovski, Sasko Ristov, and Radu Prodan. 2017. A Serverless Real-Time Data Analytics Platform for Edge Computing. *IEEE Internet Computing* 21, 4 (2017), 64–71.

Stefan Nastic, Sanjin Sehic, Duc-Hung Le, Hong-Linh Truong, and Schahram Dustdar. 2014. Provisioning Software-Defined IoT Cloud Systems. In *International Conference on Future Internet of Things and Cloud*. 288–295.

Stefan Nastic, Hong-Linh Truong, and Schahram Dustdar. 2016. A Middleware Infrastructure for Utility-based Provisioning of IoT Cloud Systems. In *IEEE/ACM Symposium on Edge Computing*. 28–40.

Suman Nath, Phillip B. Gibbons, Srinivasan Seshan, and Zachary R. Anderson. 2004. Synopsis Diffusion for Robust Aggregation in Sensor Networks. In *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*. 250–262.

Lina Ni, Jinquan Zhang, Changjun Jiang, Chungang Yan, and Kan Yu. 2017. Resource allocation strategy in fog computing based on priced timed petri nets. *ieee internet of things journal* 4, 5 (2017), 1216–1228.

Song Ningning, Gong Chao, An Xingshuo, and Zhan Qiang. 2016. Fog Computing Dynamic Load Balancing Mechanism Based on Graph Repartitioning. *China Communications* 13, 3 (2016), 156–164.

Takayuki Nishio, Ryoichi Shinkuma, Tatsuro Takahashi, and Narayan B Mandayam. 2013. Service-oriented heterogeneous resource sharing for optimizing service latency in mobile cloud. In *Proceedings of the first international workshop on Mobile cloud computing & networking*. ACM, 19–26.

Opeyemi Osanaiye, Shuo Chen, Zheng Yan, Rongxing Lu, Kim-Kwang Raymond Choo, and Mqhele Dlodlo. 2017. From cloud to Fog Computing: A Review and a Conceptual Live VM Migration Framework. *IEEE Access* 5 (2017), 8284–8300.

Claus Pahl and Brian Lee. 2015. Containers and Clusters for Edge Cloud Architectures–A Technology Review. In *2015 3rd International Conference on Future Internet of Things and Cloud (FiCloud)*. IEEE, 379–386.

Tapti Palit, Yongming Shen, and Michael Ferdman. 2016. Demystifying Cloud Benchmarking. In *IEEE International Symposium on Performance Analysis of Systems and Software*. 122–132.

John Panneerselvam, James Hardy, Lu Liu, Bo Yuan, and Nick Antonopoulos. 2016. Mobilouds: An energy efficient MCC collaborative framework with extended mobile participation for next generation networks. *IEEE Access* 4 (2016), 9129–9144.

Konstantinos E Parsopoulos, Michael N Vrahatis, and others. 2002. Particle Swarm Optimization Method for Constrained Optimization Problems. *Intelligent Technologies–Theory and Application: New Trends in Intelligent Technologies* 76, 1 (2002), 214–220.

Terry Penner, Alison Johnson, Brandon Van Slyke, Mina Guirguis, and Qijun Gu. 2014. Transient clouds: Assignment and collaborative execution of tasks on mobile devices. In *2014 IEEE Global Communications Conference*. IEEE, 2801–2806.

Deepak Puthal, Mohammad S Obaidat, Priyadarsi Nanda, Mukesh Prasad, Saraju P Mohanty, and Albert Y Zomaya. 2018. Secure and Sustainable Load Balancing of Edge Data Centers in Fog Computing. *IEEE Communications Magazine* 56, 5 (2018), 60–65.

Ramesh Rajagopalan and Pramod K Varshney. 2006. Data-aggregation Techniques in Sensor Networks: A Survey. *IEEE Communications Surveys Tutorials* 8, 4 (2006), 48–63.

Stephan Reiff-Marganiec, Marcel Tilly, and Helge Janicke. 2014. Low-latency service data aggregation using policy obligations. In *2014 IEEE International Conference on Web Services*. IEEE, 526–533.

Bhaskar Prasad Rimal, Martin Maier, and Mahadev Satyanarayanan. 2018. Experimental Testbed for Edge Computing in Fiber-Wireless Broadband Access Networks. *IEEE Communications Magazine* 56, 8 (2018), 160–167.

Sankardas Roy, Mauro Conti, Sanjeev Setia, and Sushil Jajodia. 2014. Secure data aggregation in wireless sensor networks: Filtering out the attacker's impact. *IEEE Transactions on Information Forensics and Security* 9, 4 (2014), 681–694.

Mathew Ryden, Kwangsung Oh, Abhishek Chandra, and Jon Weissman. 2014. Nebula: Distributed Edge Cloud for Data Intensive Computing. In *IEEE International Conference on Cloud Engineering*. 57–66.

Yuvraj Sahni, Jiannong Cao, Shigeng Zhang, and Lei Yang. 2017. Edge Mesh: A new paradigm to enable distributed intelligence in Internet of Things. *IEEE access* 5 (2017), 16441–16458.

Oriol Sallent, Jordi Perez-Romero, Ramon Ferrus, and Ramon Agusti. 2017. On radio access network slicing from a radio resource management perspective. *IEEE Wireless Communications* 24, 5 (2017), 166–174.

Zohreh Sanaei, Saeid Abolfazli, Abdullah Gani, and Rajkumar Buyya. 2014. Heterogeneity in mobile cloud computing: taxonomy and open challenges. *IEEE Communications Surveys & Tutorials* 16, 1 (2014), 369–392.

Daniele Santoro, Daniel Zozin, Daniele Pizzolli, Francesco De Pellegrini, and Silvio Cretti. 2017. Foggy: A Platform for Workload Orchestration in a Fog Computing Environment. In *IEEE International Conference on Cloud Computing Technology and Science*. 231–234.

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

Mahadev Satyanarayanan, Paramvir Bahl, Ramón Caceres, and Nigel Davies. 2009. The Case for VM-based Cloudlets in Mobile Computing. *IEEE pervasive Computing* 8, 4 (2009).

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

Petri Savolainen, Sumi Helal, Jukka Reitmaa, Kai Kuikkaniemi, Giulio Jacucci, Mikko Rinne, Marko Turpeinen, and Sasu Tarkoma. 2013. Spaceify: A Client-edge-server Ecosystem for Mobile Computing in Smart Spaces. In *Proceedings of the 19th Annual International Conference on Mobile Computing and Networking*. 211–214.

Omar Sefraoui, Mohammed Aissaoui, and Mohsine Eleuldj. 2012. OpenStack: Toward an Open-source Solution for Cloud Computing. *International Journal of Computer Applications* 55, 3 (2012).

Mennan Selimi, Amin M Khan, Emmanouil Dimogerontakis, Felix Freitag, and Roger Pueyo Centelles. 2015. Cloud services in the guifi. net community network. *Computer Networks* 93 (2015), 373–388.

Shashank Shekhar, Ajay Dev Chhokra, Anirban Bhattacharjee, Guillaume Aupy, and Aniruddha Gokhale. 2017. INDICES: Exploiting Edge Resources for Performance-aware Cloud Hosted Services. In *IEEE 1st International Conference on Fog and Edge Computing*. IEEE, 75–80.

Cong Shi, Vasileios Lakafosis, Mostafa H Ammar, and Ellen W Zegura. 2012. Serendipity: enabling remote computing among intermittently connected mobile devices. In *Proceedings of the thirteenth ACM international symposium on Mobile Ad Hoc Networking and Computing*. ACM, 145–154.

Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. 2016. Edge computing: Vision and challenges. *IEEE Internet of Things Journal* 3, 5 (2016), 637–646.

Pedro M Pinto Silva, Joao Rodrigues, Joaquim Silva, Rolando Martins, Luís Lopes, and Fernando Silva. 2017. Using Edge-Clouds to Reduce Load on Traditional Wifi Infrastructures and Improve Quality of Experience. In *1st International Conference on Fog and Edge Computing*. IEEE, 61–67.

Pieter Simoens, Yu Xiao, Padmanabhan Pillai, Zhuo Chen, Kiryong Ha, and Mahadev Satyanarayanan. 2013. Scalable crowd-sourcing of video from mobile devices. In *Proceeding of the 11th annual international conference on Mobile systems, applications, and services*. ACM, 139–152.

Olena Skarlat, Matteo Nardelli, Stefan Schulte, and Schahram Dustdar. 2017. Towards qos-aware fog service placement. In *2017 IEEE 1st International Conference on Fog and Edge Computing (ICFEC)*. IEEE, 89–96.

Cagatay Sonmez, Atay Ozgovde, and Cem Ersoy. 2018. EdgeCloudSim: An environment for performance evaluation of Edge Computing systems. *Transactions on Emerging Telecommunications Technologies* 29, 11 (2018), e3493.

VB Souza, Xavier Masip-Bruin, Eva Marín-Tordera, Sergio Sànchez-López, Jordi Garcia, Guang-Jie Ren, Admela Jukan, and A Juan Ferrer. 2018. Towards a proper service placement in combined Fog-to-Cloud (F2C) architectures. *Future Generation Computer Systems* 87 (2018), 1–15.

Smruthi Sridhar and Matthew E Tolentino. 2017. Evaluating Voice Interaction Pipelines at the Edge. In *2017 IEEE International Conference on Edge Computing (EDGE)*. IEEE, 248–251.

Alexandru Stanciu. 2017. Blockchain based distributed control system for edge computing. In *2017 21st International Conference on Control Systems and Computer Science (CSCS)*. IEEE, 667–671.

Ivan Stojmenovic. 2014. Fog Computing: A Cloud to the Ground Support for Smart Things and Machine-to-Machine Networks. In *Australasian Telecommunication Networks and Applications Conference*. 117–122.

Noriyuki Takahashi, Hiroyuki Tanaka, and Ryutaro Kawamura. 2015. Analysis of process assignment in multi-tier mobile cloud computing and application to edge accelerated web browsing. In *2015 3rd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering*. IEEE, 233–234.

Mohit Taneja and Alan Davy. 2017. Resource aware placement of IoT application modules in Fog-Cloud Computing Paradigm. In *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*. IEEE, 1222–1228.

Bo Tang, Zhen Chen, Gerald Hefferman, Tao Wei, Haibo He, and Qing Yang. 2015. A Hierarchical Distributed Fog Computing Architecture for Big Data Analysis in Smart cities. In *Proceedings of the ASE BigData & Social Informatics*. ACM, 28.

Jine Tang, ZhangBing Zhou, Jianwei Niu, and Qun Wang. 2013. EGF-tree: An energy efficient index tree for facilitating multi-region query aggregation in the Internet of things. In *2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing*. IEEE, 370–377.

Genc Tato, Marin Bertier, and Cédric Tedeschi. 2017. Designing Overlay Networks for Decentralized Clouds. In *2017 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*. IEEE, 391–396.

Surat Teerapittayanon, Bradley McDanel, and HT Kung. 2017. Distributed deep neural networks over the cloud, the edge and end devices. In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 328–339.

Kazuhiro Tokunaga, Kenichi Kawamura, and Naoki Takaya. 2016. High-speed uploading architecture using distributed edge servers on multi-RAT heterogeneous networks. In *2016 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN)*. IEEE, 1–2.

Nguyen B Truong, Gyu Myoung Lee, and Yacine Ghamri-Doudane. 2015. Software defined networking-based vehicular adhoc network with fog computing. In *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*. 1202–1207.

Floris Van den Abeele, Jeroen Hoebeke, Girum Ketema Teklemariam, Ingrid Moerman, and Piet Demeester. 2015. Sensor function virtualization to support distributed intelligence in the internet of things. *Wireless Personal Communications* 81, 4 (2015), 1415–1436.

Blesson Varghese, Ozgur Akgun, Ian Miguel, Long Thai, and Adam Barker. 2014. Cloud benchmarking for performance. In *2014 IEEE 6th International Conference on Cloud Computing Technology and Science*. IEEE, 535–540.

Blesson Varghese, Ozgur Akgun, Ian Miguel, Long Thai, and Adam Barker. 2016. Cloud benchmarking for maximising performance of scientific applications. *IEEE Transactions on Cloud Computing* (2016).

Blesson Varghese and Rajkumar Buyya. 2018. Next Generation Cloud Computing: New Trends and Research Directions. *Future Generation Computer Systems* 79 (2018), 849 – 861.

Blesson Varghese, Carlos Reano, and Federico Silla. 2018. Accelerator Virtualization in Fog Computing: Moving from the Cloud to the Edge. *IEEE Cloud Computing* 5, 6 (2018), 28–37.

Blesson Varghese, Lawan Thamsuhang Subba, Long Thai, and Adam Barker. 2016a. DocLite: A Docker-based lightweight cloud benchmarking tool. In *2016 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*. IEEE, 213–222.

Blesson Varghese, Nan Wang, Sakil Barbhuiya, Peter Kilpatrick, and Dimitrios S Nikolopoulos. 2016b. Challenges and opportunities in edge computing. In *2016 IEEE International Conference on Smart Cloud (SmartCloud)*. IEEE, 20–26.

Blesson Varghese, Nan Wang, Jianyu Li, and Dimitrios S. Nikolopoulos. 2017a. Edge-as-a-Service: Towards Distributed Cloud Architectures. In *International Conference on Parallel Computing (Advances in Parallel Computing)*. IOS Press, 784–793.

Blesson Varghese, Nan Wang, Dimitrios S Nikolopoulos, and Rajkumar Buyya. 2017b. Feasibility of fog computing. (2017). arXiv:1701.05451. Retrieved from https://arxiv.org/abs/1701.05451.

Stephen J Vaughan-Nichols. 2006. New approach to virtualization is a lightweight. *Computer* 39, 11 (2006).

Alexey Vinel, Jakob Breu, Tom H Luan, and Honglin Hu. 2017. Emerging Technology for 5G-enabled Vehicular Networks. *IEEE Wireless Communications* 24, 6 (2017), 12–12.

Hariharasudhan Viswanathan, Eun Kyung Lee, and Dario Pompili. 2016. A Multi-Objective Approach to Real-Time In-Situ Processing of Mobile-Application Workflows. *IEEE Transactions on Parallel and Distributed Systems* 27, 11 (2016), 3116–3130.

Michael Vögler, Johannes Schleicher, Christian Inzinger, Stefan Nastic, Sanjin Sehic, and Schahram Dustdar. 2015. LEONORE–Large-Scale Provisioning of Resource-constrained IoT Deployments. In *IEEE Symposium on Service-Oriented System Engineering*. 78–87.

Thang X Vu, Symeon Chatzinotas, and B Ottersten. 2017. Edge-caching wireless networks: Energy-efficient design and optimization. *CoRR* (2017).

Huaqun Wang, Zhiwei Wang, and Josep Domingo-Ferrer. 2018. Anonymous and Secure Aggregation Scheme in Fog-based Public Cloud Computing. *Future Generation Computer Systems* 78 (2018), 712 – 719.

Jianyu Wang, Jianli Pan, and Flavio Esposito. 2017a. Elastic Urban Video Surveillance System Using Edge Computing. In *Proceedings of the Workshop on Smart Internet of Things*. 7.

Nan Wang, Blesson Varghese, Michail Matthaiou, and Dimitrios S Nikolopoulos. 2017c. ENORM: A framework for edge node resource management. *IEEE transactions on services computing* (2017).

Shiqiang Wang, Rahul Urgaonkar, Ting He, Kevin Chan, Murtaza Zafer, and Kin K Leung. 2017b. Dynamic service placement for mobile micro-clouds with predicted future costs. *IEEE Transactions on Parallel and Distributed Systems* 28, 4 (2017), 1002–1016.

Yu Xiao, Marius Noreikis, and Antti Ylä-Jaäiski. 2017. Qos-oriented capacity planning for edge computing. In *2017 IEEE International Conference on Communications (ICC)*. IEEE, 1–6.

Qichao Xu, Zhou Su, Qinghua Zheng, Minnan Luo, and Bo Dong. 2018. Secure content delivery with edge nodes to save caching resources for mobile users in green cities. *IEEE Transactions on Industrial Informatics* 14, 6 (2018), 2550–2559.

Yiming Xu, V Mahendran, and Sridhar Radhakrishnan. 2016. Towards SDN-based fog computing: MQTT broker virtualization for effective and reliable delivery. In *2016 8th International Conference on Communication Systems and Networks (COMSNETS)*. IEEE, 1–6.

Binxu Yang, Wei Koong Chai, George Pavlou, and Konstantinos V Katsaros. 2016. Seamless Support of Low Latency Mobile Applications with NFV-Enabled Mobile Edge-Cloud. In *2016 5th IEEE International Conference on Cloud Networking (Cloudnet)*. IEEE, 136–141.

Binxu Yang, Wei Koong Chai, Zichuan Xu, Konstantinos V Katsaros, and George Pavlou. 2018. Cost-Efficient NFV-Enabled Mobile Edge-Cloud for Low Latency Mobile Applications. *IEEE Transactions on Network and Service Management* (2018).

Ashkan Yousefpour, Caleb Fung, Tam Nguyen, Krishna Kadiyala, Fatemeh Jalali, Amirreza Niakanlahiji, Jian Kong, and Jason P Jue. 2019. All one needs to know about fog computing and related edge computing paradigms: a complete survey. *Journal of Systems Architecture* (2019).

Fei Yuan, Yiju Zhan, and Yonghua Wang. 2014. Data density correlation degree clustering method for data aggregation in WSN. *IEEE Sensors Journal* 14, 4 (2014), 1089–1098.

Matei Zaharia, Tathagata Das, Haoyuan Li, Timothy Hunter, Scott Shenker, and Ion Stoica. 2013. Discretized streams: Fault-tolerant streaming computation at scale. In *Proceedings of the twenty-fourth ACM symposium on operating systems principles*. ACM, 423–438.

Engin Zeydan, Ejder Bastug, Mehdi Bennis, Manhal Abdel Kader, Ilyas Alper Karatepe, Ahmet Salih Er, and Mérouane Debbah. 2016. Big data caching for networking: Moving from cloud to edge. *IEEE Communications Magazine* 54, 9 (2016), 36–42.

Ning Zhang, Peng Yang, Shan Zhang, Dajiang Chen, Weihua Zhuang, Ben Liang, and Xuemin Sherman Shen. 2017. Software defined networking enabled wireless network virtualization: Challenges and solutions. *IEEE Network* 31, 5 (2017), 42–49.