Open Access

Open Journal of Internet Of Things (OJIOT)
Volume 5, Issue 1, 2019

www.ronpub.com/ojiot
ISSN 2364-7108

Data-Centric Resource Management in Edge-Cloud Systems for the IoT

Igor Leão dos Santos[A], Flávia C. Delicato[B], Paulo F. Pires [B], Marcelo Pitanga Alves[C], Ana Oliveira[D], Tiago Salviano Calmon[D]

[A] Programa de Pós-graduação em Engenharia de Produção e Sistemas (PPPRO), Centro Federal de Educação Tecnológica Celso Suckow da Fonseca (CEFET-RJ), Rio de Janeiro, Brazil, igor.santos@cefet-rj.br
[B] Programa de Engenharia de Sistemas e Computação (PESC), Universidade Federal do Rio de Janeiro (UFRJ), Rio de Janeiro, Brazil, paulo.f.pires@gmail.com, fdelicato@dcc.ufrj.br
[C] Programa de Pós-Graduação em Informática (PPGI), Universidade Federal do Rio de Janeiro (UFRJ), Rio de Janeiro, Brazil, mpitanga@gmail.com
[D] Dell EMC Brazil, Rio de Janeiro, Brazil, {ana.oliveira,tiago.calmon}@dell.com

ABSTRACT

A major challenge in emergent scenarios such as the Cloud-assisted Internet of Things is efficiently managing the resources involved in the system while meeting requirements of applications. From the acquisition of physical data to its transformation into valuable services or information, several steps must be performed, involving the various players in such a complex ecosystem. Support for decentralized data processing on IoT devices and other devices near the edge of the network, in combination with the benefits of cloud technologies has been identified as a promising approach to reduce communication overhead, thus reducing delay for time sensitive IoT applications. The interplay of IoT, edge and cloud to achieve the final goal of producing useful information and value-added services to end user gives rise to a management problem that needs to be wisely tackled. The goal of this work is to propose a novel resource management framework for edge-cloud systems that supports heterogeneity of both devices and application requirements. The framework aims to promote the efficient usage of the system resources while leveraging the Edge Computing features, to meet the low latency requirements of emergent IoT applications. The proposed framework encompasses (i) a lightweight and data-centric virtualization model for edge devices, (ii) a set of components responsible for the resource management and the provisioning of services from the virtualized edge-cloud resources.

TYPE OF PAPER AND KEYWORDS

Regular research paper: *Resource management, Edge computing, Cloud-assisted IoT, Virtualization model*

1 INTRODUCTION

Cloud computing technology has revolutionized the way end-users and enterprises gain access to computing resources, enabling the on-demand allocation and release of a wide range of services and resources. The flexibility and business model provided by the cloud

This paper is accepted at the *International Workshop on Very Large Internet of Things (VLIoT 2019)* in conjunction with the VLDB 2019 conference in Los Angeles, USA. The proceedings of VLIoT@VLDB 2019 are published in the Open Journal of Internet of Things (OJIOT) as special issue.

computing make this paradigm very appealing and enable novel applications. Another technological trend that has been gaining momentum recently is the Internet of Things (IoT) [1], which enables the interconnection with the Internet of the most varied physical objects, instrumented by intelligent sensors and actuators. With the possibility of addressing each physical object individually and making it part of a global network, the IoT has the potential to provide novel applications to make life easier and healthier for citizens, to increase the productivity of companies and to promote the building of more intelligent and sustainable cities, environments and countries.

A key challenge in IoT is efficiently managing the system resources. IoT devices, such as sensor devices, have limited computing and energy resources, and thus are not able to perform sophisticated processing and storing large amounts of data. Therefore, it is often necessary to rely on more powerful devices to fully perform the transformation process required by IoT applications [9]. With its vast capacity of processing and long-term storage, cloud computing is an appealing platform to be combined with IoT to create complex, large-scale, distributed, and data-oriented ecosystems. However, some features of cloud computing make it unsuitable to meet requirements of IoT applications.

The essentially centralized nature of the cloud does not fit well with the inherently decentralized nature of IoT. In IoT, data is often generated from geographically distributed sources, and can be consumed by equally dispersed users, often using devices that themselves are also part of IoT. Blindly sending this distributed data for processing and storage centrally in the cloud, then forwarding it back to users near data sources, can result in unwanted delays. For some applications, response time is a critical quality requirement, and the latency and unpredictability of communication with the cloud can lead to performance degradation.

Support for decentralized data processing on devices near the edge of the network, in combination with the benefits of cloud technologies has been identified as a promising approach to reduce communication overhead and data transfer time (hence the latency for applications). In this context, the conceptual approach known as Fog [3] or Edge Computing [30] has emerged, which advocates moving part of the computing and storage resources closer to the edge of the network, in a decentralized way.

Physical edge devices are heterogeneous in terms of their capabilities and can be either resource-poor devices such as access points, routers, switches, base stations, and smart sensors, or resource-rich machines like a “cloud-in-a-box”, or Cloudlets [27]. Edge devices may perform several tasks, such as data preprocessing and filtering, reconstructing raw data into a more useful form, uploading only the necessary data to the cloud. In addition, edge nodes can monitor smart objects and sensors activities, keeping check on their energy consumption. The edge consumes locally the portion of data generated by sensors that require real-time processing (from milliseconds to tenths of seconds). Then, it transmits the rest of such data to the cloud, for operations with less stringent time constraints (from seconds to minutes). Therefore, edge computing allows delivery of data with low latency. On the other hand, the closer to the cloud, the longer the time scale, and the wider is the geographical coverage. The cloud provides the ultimate and global coverage and serves as a repository for data for the duration of months or years, besides allowing more complex data analytics.

The interplay of IoT, edge and cloud to achieve the final goal of producing useful information and value-added services to end users gives rise to a management problem that needs to be wisely tackled. Both cloud and edge computing strongly build on the virtualization concept. However, virtualization of devices at the edge needs to follow a lighter and more flexible approach to meet the constraints and heterogeneity of devices and exploit their specific features. The authors in [19] claim that to fully achieve the potential of edge computing for IoT, four concerns need to be addressed: abstraction, programmability, interoperability, and elasticity. In particular for a three-tier IoT-edge-cloud architecture, it is crucial to provide simple and yet efficient configuration and instantiation methods that are independent of the technologies used by different IoT and cloud providers.

We propose a novel resource management framework for edge-cloud systems that supports heterogeneity of both devices and application requirements. The framework promotes the efficient usage of the system resources while leveraging the Edge Computing features, exploring the advantages of service provision at the edge of the network, to meet the low latency requirements of emergent applications. The framework encompasses (i) a lightweight and data-centric virtualization model for edge devices, (ii) a set of components responsible for the resource management and the provisioning of services from the virtualized edge-cloud resources.

2 CHALLENGES IN EDGE-CLOUD ECOSYSTEMS

Considering a heterogeneous edge-cloud ecosystem, built to serve multiple applications with different requirements, the need arises to provide a framework to manage the available resources in an efficient and cost-effective way. The core issue of this problem is how to allocate the resources available in the heterogeneous edge-cloud system in order to accommodate the

requirements posed by multiple applications. At first glance, this issue is similar to the typical resource allocation problem, which has been exhaustively studied in several areas of computing systems. However, resource allocation for edge-cloud systems poses new challenges that call for novel solutions, tailored for such an emerging scenario. Examples of specific features are the huge heterogeneity of the participant devices (from tiny sensors to middle-tier gateways to powerful data center nodes), the highly dynamic execution environment, and the nature of the data generated by IoT devices.

The complexity in the development of solutions for resource allocation in edge-cloud attract the attention of researchers in search of efficient computational solutions to meet the requirements of emerging applications (e.g., low latency, mobility, energy efficiency, scalability, etc.) envisioned to execute on such scenarios [3][36]. Solutions for resource management, including resource allocation and provisioning, are well established in the Cloud computing field. However, in the context of Edge and Fog computing, there are still many open issues in this regard [25][10][35]. According to [35], there are no distributed computing frameworks that fully and properly manage edge node resources.

We claim that resource management is a key issue to deal with the diverse nature of resources encompassed in an edge-cloud system and to optimize the overall system performance. Providing effective solutions to this challenge will bring benefits on one hand, to end users and on the other hand, to infrastructure providers and device owners. In this sense, we propose a novel approach for resource management in edge-cloud systems.

2.1 Heterogeneous Devices

In the edge-cloud environment, multiple devices with different processing capabilities exist and can collaborate to meet the applications' goals and requirements. Powerful computers such as the ones hosted in the cloud can rely on legacy virtualization technologies without major issues, but devices in the lower tiers might get their performance impacted critically with these technologies. It is important, then, to consider the heterogeneity of devices in the design of the virtualization engine. Specifically, the resource constrained nature of several types of devices at the edge tier needs to be taken into account in any solution for virtualization and resource management. Due to resource constraints from edge devices compared to data centers in the cloud, multiple edge devices often need to somehow collaborate so as to accomplish intensive application tasks by sharing the workload between them. The resource management framework, supported by its virtualization model, must enable such collaboration in a natural way.

2.2 Heterogeneous Applications

In addition to the high heterogeneity of devices, multiple applications with different functional and non-functional (QoS-related) requirements can co-exist using resources from the same underlying infrastructure. Some applications might be more computationally intensive, whereas others might have low latency requirements, for example. Moreover, several applications have severe restrictions on data security. Data generated by users' devices often contain personal information, such as photos/videos taken by mobile phones, GPS information on the user location, health information sensed by wearable devices, and smart home status. Processing and storage of sensitive data must be handled carefully to avoid privacy issues. The decision of placing a given service in one computational node (located at the edge or the cloud for instance) must consider the requirements of the specific applications the node is serving. A resource management framework must be able to handle different kinds of applications with different (and sometimes even conflicting) requirements.

2.3 Ultra-Large Scale

Edge-cloud ecosystems are complex environments encompassing many heterogeneous components. One major component is the myriad of devices acting as data sources. Considering the increasing availability of smart sensors, mobile phones, wearable and other IoT devices, the resulting system may encompass hundreds to millions of connected devices, producing a massive amount of data to be processed and stored. Therefore, any solution for resource management must be scalable in terms of the number of computational nodes and the number of application requests to be served. The ultra large scale of systems brings several challenges mainly regarding the coordination of the nodes actively engaged in providing the required resources to meet the application requests. It is important to mention that several authors (as [7]) point out that a considerable deficiency in current works in edge computing is the lack of support for collaborative computation.

2.4 Data-Centric Nature

The current overabundance of data, generated by various emerging applications as social media and IoT, has caused several changes in how such data should be

processed and stored. Data generated by embedded sensors and applications running on mobile devices in users' personal space may not necessarily be sent blindly to the remote cloud. There are new demands to shepherd data within and across multiple tiers from the edge of the network, through the core to the super data centers in the cloud. Data may be shared, (pre)processed and cached in local and/or edge nodes and then may transit to other tiers of the infrastructure while being used, reused, combined and re-purposed to derive value-added information, analytical insights *en route* to being consumed and possibly archived [28].

Processing in the multiple tiers of an edge-cloud system needs to take advantage of node heterogeneity, take into account the dynamism of the environment, and also needs to consider the data content in decision-making tasks. The execution of application-specific functions, data fusion procedures, and knowledge extraction can occur at various points along the path between the data source and the cloud. Sometimes results can be taken en route, without even requiring additional forwarding to the cloud. For this, the content of the data has fundamental value in decision-making and intermediate processing. Furthermore, a piece of data might be re-used by several applications in different contexts, placed in different nodes.

In short, we argue that the data needs to be raised to first-class citizens in these ecosystems. Therefore, virtualization solutions for such environments must be data-centric. Not only features like Virtual Machines (VMs) and processing cores, commonly used in traditional virtualization models, but the data itself, its metadata and handling functions, need to be virtualized. Moreover, VMs created with this data-centric view should be placed on distributed physical nodes across multiple tiers, not only at the cloud.

3 A NOVEL RESOURCE MANAGEMENT FRAMEWORK FOR EDGE-CLOUD SYSTEMS

To address the aforementioned challenges, our proposal comprises a software framework encompassing (i) a light data-centric virtualization model for edge-cloud systems, and (ii) a set of software components responsible for the resource management and the provisioning of services using the virtualized edge-cloud resources.

3.1 LW-Dc4EC (Light Weight Data-Centric Model for Edge-Cloud)

In this section, we detail LW-Dc4EC, our novel data-centric virtualization model for edge-cloud systems. Its goal is to offer a lightweight virtualization on top of physical sensor and actuator nodes (here denoted as PSAN), of Edge nodes (EN) and of Cloud Nodes (CN). This model is supported by a three-tier architecture for edge-cloud systems (shown in Figure 1). LW-Dc4EC uses a set of different techniques for the creation of virtual nodes. Six built-in, predefined types of virtual nodes are initially provided (explained later and depicted in Figure 2). However, since LW-Dc4EC was conceived with extensibility in mind, new types can be defined and easily incorporated in the model. A new type is created by extending a virtual node super-class available into the framework core library and template files to configure the desired data type.

3.1.1 The Three-Tier Architecture

Figure 1 illustrates the architecture to support the proposed virtualization model. It is composed of three tiers: (i) Cloud tier (CT), (ii) Edge tier (ET), and (iii) Sensor (or data source) tier (ST).

The Edge tier (ET) encompasses the edge nodes (EN) whereas the Cloud tier (CT) encompasses the cloud nodes (CN). Both tiers host the physical devices of the ET and CT, respectively. The EN and CN are virtualized by using traditional models for cloud and edge virtualization. They have properties such as processing speed, total memory, bandwidth and geographical location. However, there are some important differences between ENs and CNs. ENs are less powerful devices than CNs, regarding the resources available (e.g., memory capacity). Besides, they are geographically closer to the data sources (for instance sensors and IoT devices) than CNs. Another difference is the centralized nature of the nodes at the Cloud tier, while edge nodes are typically decentralized entities and may leverage distributed and collaborative computing. The distributed nature and the proximity of the data sources make it possible to exploit context and location-awareness capabilities in the edge tier. Thus, instead of providing resources from a centralized and remote infrastructure, one can explore the provision of resources regionally distributed, either closer to the data source, the data consumer, or both. This feature has the potential to increase the efficiency of the usage of the infrastructure and the quality of the user experience with the services provided.

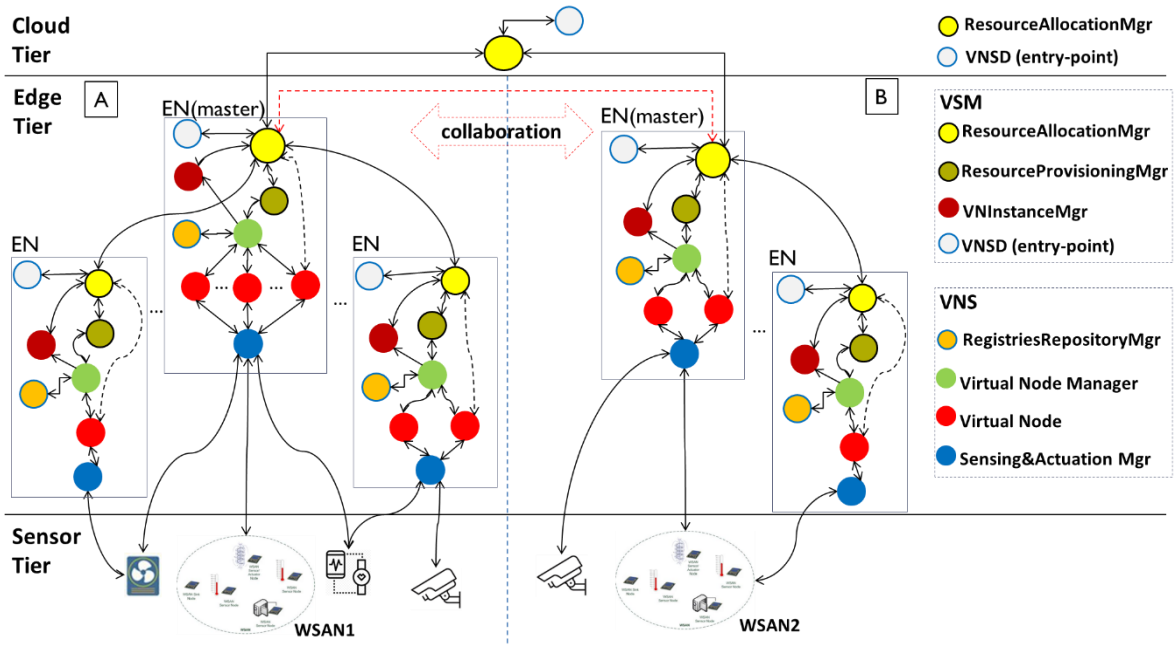

Figure 1: Three-tier architecture

In our architecture, we actively promote the collaboration among edge nodes and the location-awareness features. The nodes in the Edge Tier are grouped in a hierarchical fashion, so that we have both vertical and horizontal communication/collaboration within the system [17]. To reach this goal, firstly, we created hierarchical groups of edge nodes using an appropriate hierarchy algorithm to promote vertical communication/collaboration. In our solution, the horizontal communication/collaboration only occurs between the master edges nodes of each hierarchy. Thus, we used the Weighted Voronoi Diagram (WVD) [14] as a solution to build neighborhoods of the master edge nodes in order to promote collaboration between them. The WVD uses "sites" (geographic locations) on a map to divide it into regions. Therefore, to use the WVD algorithm, we need to provide the master edge nodes geographic locations.

In the created hierarchy, the master nodes are responsible for engaging slave edge nodes to serve an application request. We also organize the master edge nodes in a neighborhood, in order to enable the collaboration among them. Thus, the master edge nodes can perform a collaboration process with each other to identify a group of edge nodes that can serve the application request. With such hierarchical and geographical organization of the nodes, it is possible (i) to facilitate the collaboration between the edge nodes, (ii) to assign physical nodes (at the Sensor Tier) to edge nodes that are closer to them, thus minimizing the consumption of resources with data and control messages, since we keep the communications within a limited geographic region.

Finally, the Sensors Tier (ST) encompasses a set of constrained end devices deployed over a geographic area that consist the data sources for the edge-cloud system. Each device is heterogeneous regarding its processing speed, total memory, and energy capacity. Besides, end devices at this tier have the capacity of providing sensing data and/or performing actuation tasks over a region. Examples of devices are wireless sensors grouped to compose Wireless Sensor and Actuator Networks (WSANs), and smart devices such as smart phones, smartwatches, etc.

In the considered architecture, we assume that the CN is responsible for hosting the Virtual Node Service Delivery (VNSD). It is an entry point to receive the user requests. In addition, the CN is responsible for hosting a centralized version of the Resource Allocation process. The edge nodes (ENs) provide the major computational units organized in two subsystems, namely Virtualization Subsystem Manager (VSM) and Virtual Node subsystem (VNS). The VSM encompasses the ResourceAllocationMgr, ResourceProvisioningMgr, VNInstanceMgr (VNIR) and Virtual Node Service Delivery (VNSD) whereas the VNS includes the RegistriesRepositoryMgr, Virtual Node Manager, Virtual Node and Sensing&ActuationMgr. These units are responsible for handling the user requests by performing tasks to either provide sensing data or perform actuations on the physical environment.

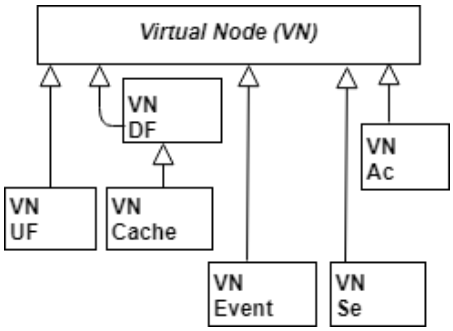

Figure 2: Types of VNs

Table 1: Types of VNs

Parameters	Description
UF	User function
Se	Sensing
Ac	Actuation
Df	Data function
Ch	Cache
Ev	Event

3.1.2 Virtualization Model

The concept of virtualization is commonly adopted to hide heterogeneity and complexity of resources to be provided, thus facilitating their management and utilization. The core idea of virtualization in an edge-cloud system is to abstract away "physical resources", which can then be "composed" at a logical level to support usage by multiple independent users and even by multiple concurrent applications.

As traditional cloud platforms, edge computing is also strongly built on the virtualization concept. However, virtualization of resources at the edge tier needs to follow a lighter and more flexible approach to meet the constraints and heterogeneity of devices and to exploit the specific features of these nodes. Moreover, for emerging applications as IoT, besides processing, storage and bandwidth capacities, and an extremely valuable resource is the sensing data produced by the IoT devices. Therefore, first-order entities in a virtualization process are no longer just virtual machines and computational cores, but also sensing data (raw or in different processing states). An edge-cloud virtualization model that addresses such applications needs to consider this data-driven nature as well.

To meet the requirements of being lightweight, the proposed virtualization model is based on microservices and container technology. More specifically, for the specification of our virtualization model, we adopted an approach based on microservices [11][20] and for the implementation of this model we propose adopting a container-based solution [22][15].

Microservices are small, highly decoupled applications, built on a single responsibility. They are independently deployable, scalable, and testable and they communicate with each other using well defined application programming interfaces (API). In turn, the container-based approach can be defined as a lightweight virtualization technology for packaging, delivering and orchestrating both software infrastructure services and applications, aiming at increasing interoperability and portability.

The motivation for using microservices in the context of this work is to allow the development of independent and lightweight components for running on the edge nodes. We use containers to package such components in lightweight images, thus facilitating their distribution and managing. Another relevant feature of containers is to facilitate their migration between computational nodes, in the context of this work, between edge nodes [33]. Component migration is an important feature for many applications, mainly in the presence of mobile nodes, since the edge nodes serving an application running in the mobile device may become too far to meet the required delay.

To meet the requirement of being data-oriented, and thus more tailored for IoT applications, data is the core entity for creating the virtual nodes in the proposed virtualization model. We defined several types of virtual nodes that represent data-driven resources to be provided by the edge-cloud infrastructure. Applications access the resources provided by our three-tier architecture through the virtualization model.

The virtual node (VN) is the central element of the model in LW-Dc4EC. The VN (Figure 2) is a software instance providing data in response to application requests directly at the edge of the network. It is responsible for abstracting the computation and communication capabilities provided by a set of underlying nodes. Moreover, our VN is based on the microservice concept, as it is small, highly decoupled, and performs a single responsibility. Thus, each virtual node is designed to implement one data type. Therefore, our model already provides predefined types of VNs for each one data type provided (Table 1).

A virtual node is formally defined as a tuple VN = (RS, GL, NT), where RS represents the resource provided by the VN; GL = (longitude, latitude) is the geographic location of interest; and NT = {UF, Se, Ac, DF, Ch, Ev} is a collection of VN types (Table 1). Resources can be of simple type such as Temperature or a complex type, such as the description of an event

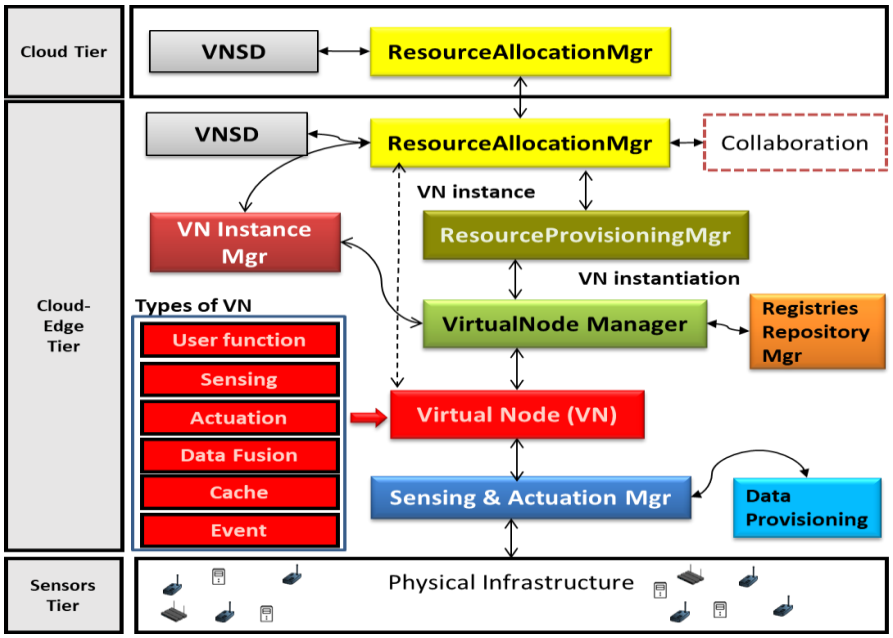

Figure 3: Components for resource management in edge-cloud systems

of interest (as Fire Detection, Fire Intrusion, Rain, Target Detected, etc.). Hereafter, we describe each type of VN. The VN of type user function (UF) allows the user to inject code for performing custom operations (application specific) over data.

The VN of type sensing (Se) provides a stream of raw data sensed from the physical environment and has a set of properties p: p = (fr, sr), where fr denotes the data freshness and sr the sampling data rate. The data stream can be retrieved from historical databases maintained at the edge tier or by a direct connection with the physical nodes at the sensor tier. The data freshness [4] is an important requirement that a VN must verify during the processing of the request to determine the source of the data to send to the application. For instance, if the last data delivered is in a valid range time of data freshness, the VN transmits the data obtained from the cache to the application. Otherwise, a fresh data is gotten using the Sensing & Actuation sub-process before forwarding it to the application.

The VN of type actuation (Ac) provides actuation capabilities over the physical environment and has a set of properties p: p = (op, tx), where op denotes the type of actuation function provided by the VN and tx is the frequency that the actuation command must be performed by the system.

The VN of type data fusion (DF) provides value-added information through the execution of queries using a Complex Event Processing (CEP) engine [8] and has a set of properties p: p = (af, sn), where af denotes an information/aggregation function and sn the number of samples to be used by af. This is a very powerful type of VN since it allows defining application-specific functions, domain-specific functions or generic event processing functions.

The VN of type cache (Ch) is a subtype of DF that adds the capability of persisting the results of af in memory. The VN Ch has a set of properties p: p = (ts), where ts denotes the timestamp of the execution of af (that produced the data cached by VN Ch). This VN is important to avoid unnecessary use of resources of an EN when several requests are received for processing the same query using the same parameters.

Finally, the VN of type event (Ev) aims to notify an application or another VN whenever an event of interest occurs by using a publish/subscribe communication model [2][34]. VN Ev has a set of properties p: p = (rl), where rl denotes a rule to trigger the event.

3.2 Resource Management Framework

The Resource Management activity in cloud computing encompasses the resource allocation and resource provisioning, among other activities. These are two key processes and planned to ensure the operational efficiency of the entire cloud system. Proper resource allocation improves overall performance of the system and avoids different kinds of bottlenecks, that could otherwise degrade performance of the running applications.

In this context, we propose a novel approach for resource management in edge-cloud systems. An innovative aspect of our proposal is to consider in an integrated way the virtualization and the resource

management processes. We believe that, since edge-cloud ecosystems essentially provides virtualized resources, the efficient and cost-effective provisioning and allocation of such resources are intrinsically entangled with the virtualization process itself. Therefore, our resource management framework provides a set of components and respective activities for the instantiation of VNs that encompass the processes for (i) the resource allocation, (ii) the resource provisioning, (iii) managing sensing & actuation tasks (required for task scheduling), (v) data provisioning, and (vi) collaboration process. Figure 3 summarizes the relation among the components in charge of performing such processes. In the following we briefly describe each component in the context of the edge-cloud infrastructure operational flow.

End users submit their requests to the edge-cloud system using an API deployed at the Cloud or via an Edge node. The arriving requests are handled by the ResourceAllocationMgr (RA) component responsible for implementing the Resource Allocation process (described in the next section). When requests arrive via Cloud, a centralized version of the RA component is responsible for forwarding each request to the master edge node (EN) capable of meeting it. Upon receiving the requests, the RA executing in the EN must provide a VN instance to meet such requests. To do so, the RA component searches in its cache of VN instances and queries all its available slave nodes by a VN matching the received requests. When a matching VN is found, the RA component forwards the request for the VN to execute the tasks thus providing the requested data/event as outcome. However, if a VN is not found or if the available VNs are busy (with other, previously received) then the ResourceProvisioningMgr component is invoked.

The ResourceProvisioningMgr (RP) is the component in charge of executing the action to select and prepare the underlying physical infrastructure that is capable of hosting and running a VN instance (a container in our proposal) matching application requests. The action of selecting physical nodes that meet the requirements of data provisioning to compose a virtual node is a mapping function, for which there are some proposals in recent literature [32][18][6][23][31].

In our proposed framework, in order to provision a VN, the RP component invokes the VirtualNode Manager (VNM) component. The VNM is an auxiliary component in charge of instantiating the appropriate type of VN to meet the application request, besides registering the new VN instance into the instance repository. However, if the RP is not capable to provide the necessary resources to instantiate a VN, the following operational decisions are executed:

- If the EN is a slave node and the request has arrived directly by the VNSD (entry-point), the request is forwarded to its respective master node;
- If the EN is a master node and the request has arrived by the point of entry or forwarded by a slave node, the master node invokes the collaboration process to find a neighbor node and then, forwards the request to the neighbor master node. Whenever the collaboration process is not able to find a neighbor master node to meet the request, then it is forwarded to the centralized Resource AllocationMgr component at the Cloud.

Collaboration is the process responsible for enabling the cooperative work and the division of the workload to meet an application request among the edge nodes. This process is available (deployed) into all the edge nodes, but only the edge nodes classified into the hierarchy as Masters are in charge of executing the collaboration. Thus, the collaboration process provides for each master edge node the capability of decision-making to engage neighboring master edge nodes to allocate or provision VNs whenever it is necessary.

Several authors (as [7]) identify a lack of support for collaborative computation in edge-cloud systems. That is, existing approaches do not seem to consider situations when multiple edge devices can somehow collaborate to accomplish an intensive task by sharing the workload between them. In this sense, the proposed framework fills a research gap by providing mechanisms and building blocks to promote collaboration between edge nodes.

When a VN receives a request from the RA to process, it uses the services of the Sensing & Actuation Mgr. It is the component implementing the process in charge of managing all interactions between the VN and the physical environment, i.e., the Sensor (data source) Tier (ST). It is an independent component that continuously gets data/events from the physical devices and persists them into the historical database maintained at the Edge tier. Its importance is to abstract the complexity of the VN to deal with the highly heterogeneous devices that directly get data/perform actuations from/upon the physical environment. Therefore, the SA provides the services for the VN to acquire sensing data and/or send actuation commands (depending on the type of VN).

The provided data can be either preprocessed or unprocessed. Unprocessed data are retrieved from historical databases or by directly accessing the physical nodes at the sensor tier whenever fresh data is required. The processed data are provided by a Complex Event Processing (CEP) engine. The CEP [8] engine is responsible for the execution of queries that making use of single or a set of raw data as input.

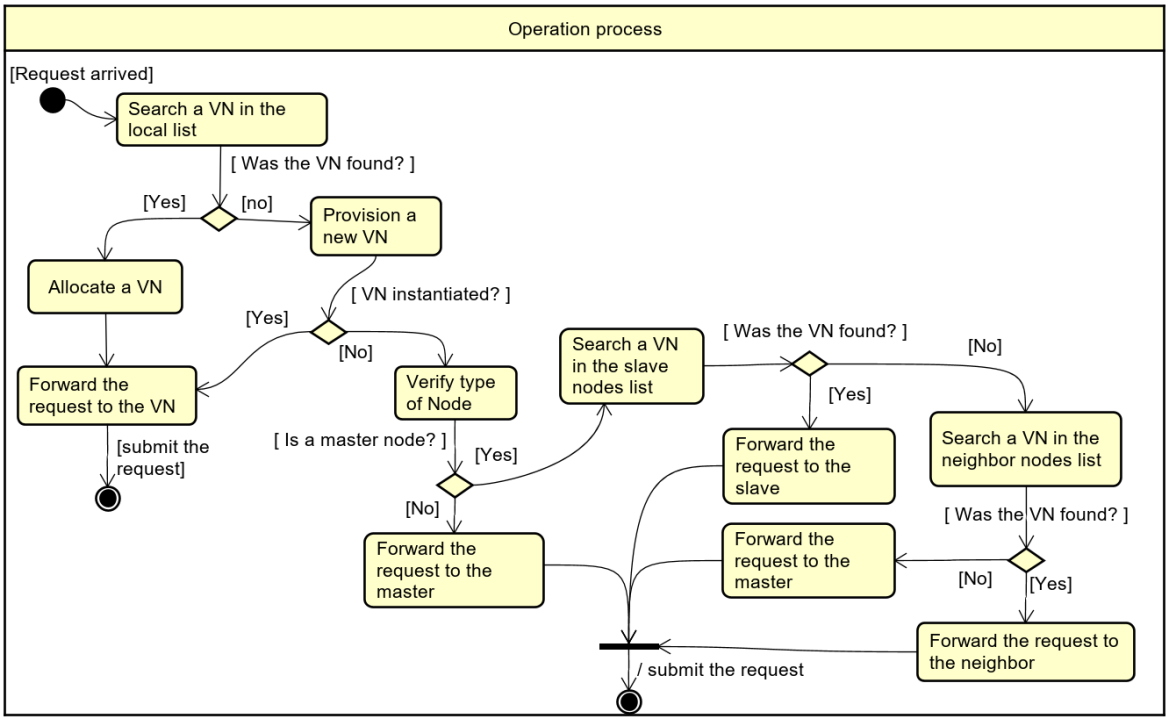

Figure 4: Overview of the Resource Management operation process

Furthermore, the SA services providing data to VNs make use of the Data provisioning process. The Data provisioning process is responsible for abstracting the complexity of dealing with operations for the data collects from the physical devices, data persistence, data update, data delete, and data retrieval in the historical databases.

Figure 4 summarizes the operational flow of the proposed framework. In the following section, we will detail the components responsible for performing the processes covered by the resource management framework.

3.2.1 The Software Components and their Behavioral View

Figure 5 illustrates the framework components, their services, and relationships, as well as the tier in which they are deployed, considering the 2 upper tiers (Cloud and Edge) of our 3-tier architecture. The Edge Tier hosts two subsystems, the Virtual Node Subsystem (VNS) and the Virtualization Subsystem Manager (VSM).

The SAM provides connectors for abstracting the heterogeneity of the physical objects/devices and allowing the interaction with them. Devices or IoT objects include but are not limited to smart sensors of several types, home appliances, alarm systems, heating, and air conditioning, lighting, industrial machines, irrigation systems, drones, traffic signals, automated transportation, and so forth. The connector is a component that encompasses (i) a driver interface responsible for interaction with the physical device, (ii) services for data transformations, and (iii) handlers for servicing requests.

The DSM component is responsible for storing the data in the temporary database at the edge nodes, besides providing the basic operations for persistence, update, delete and retrieval data. VN is an abstraction used to design six predefined types of VN components (VNSe, VNac, VNdf, VNuf, VNcache, and VNevent) to handle the application requests. The VN exposes the IVN interface to provide its services. It should be mentioned that, in our view of an edge-cloud system, the infrastructure provider will offer its services through formal or semi-formal agreements with users. Therefore, a predefined set of virtual nodes can already be provided a priori to meet the potential applications domains or specific applications whose contracts have already been established.

Some applications' requirements may be met by the services of a single type of virtual node while others will require combined services of multiple types. As the envisaged edge-cloud scenario is dynamic, applications may eventually arrive with requirements that are not met by the original set of VN types. Such applications may require the specification of new types, which will be

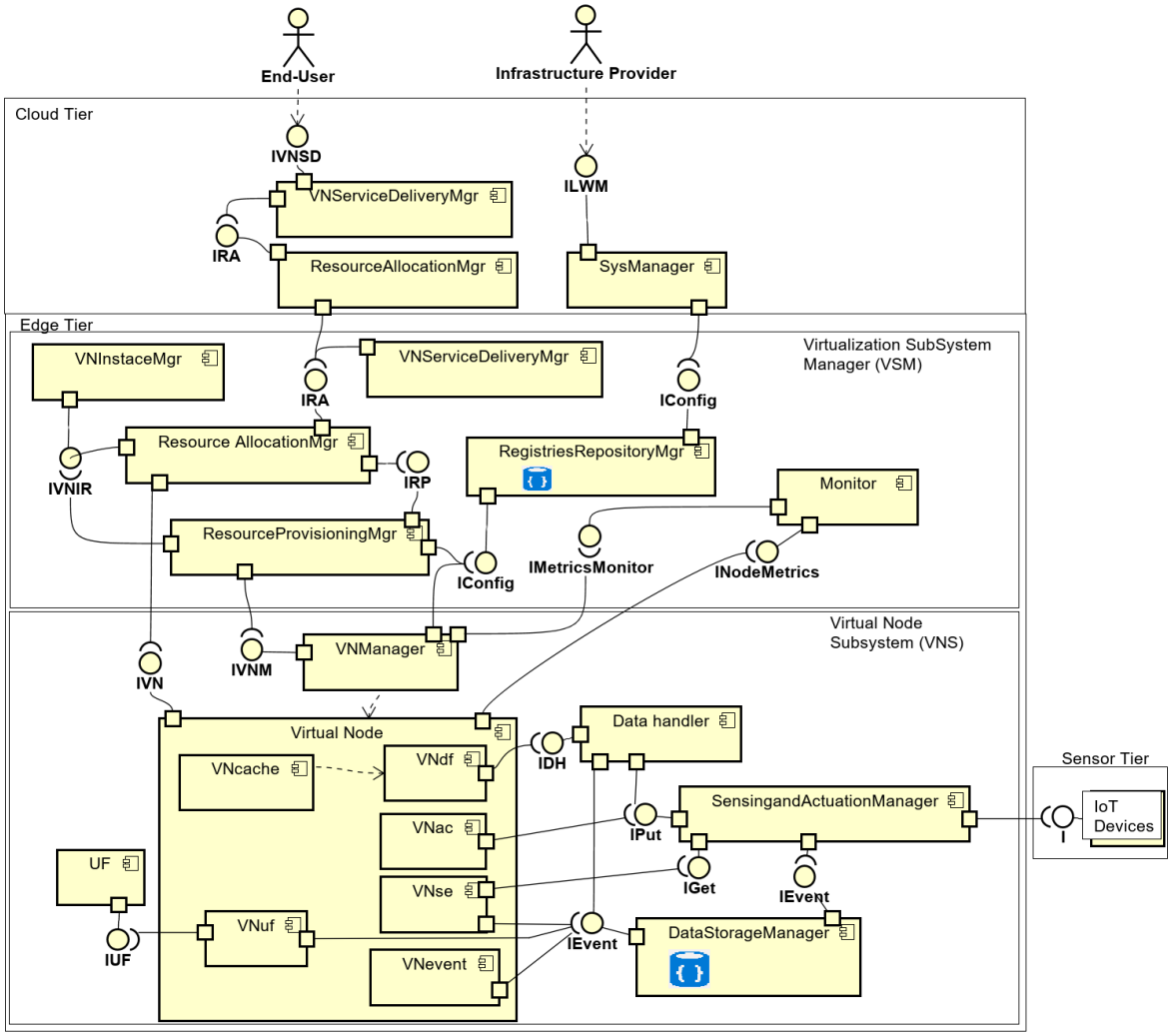

Figure 5: Framework software components

extensions of the existing ones. UF is the component that hosts the user defined functions. The DH component is responsible for abstracting the complexity of executing queries over sensed data.

The Virtualization Subsystem Manager (VSM) encompasses six components: VNServiceDeliveryMgr (VNSDM); ResourceAllocatorMgr (RAM); VNIntanceMgr (VNIR); ResourceProvisioningMgr (RPM); RegistriesRepositoryMgr (RR), Monitor. The VNSDM goal is to receive requests that arrive at the system. Since requests can enter the system via Edge and Cloud tiers, this component is deployed in both tiers. VNSDM offers a set of APIs through the IVNSD interface to allow users: (i) request data/events to the system, (ii) send an actuation command to the VN for execution, and (iii) discover registered VNs. The component RAM is in charge of implementing the algorithm that allocates instances of Virtual Nodes (VNs) to meet application requests. It offers the ResourceAllocatorInterface (IRA) used to receive the requests arriving via the VNSDM or forwarded by the centralized RAM.

The VNIR is the component responsible for managing a pool of VN instances in memory. RPM is the component in charge of provisioning a new VN instance whenever it is necessary. It provides its service through the ResourceProvisionerInterface IRP. The component RR is responsible for providing the services to store, remove, and retrieve metadata related to the data types registered into the system by the Infrastructure Provider (InP). Its services are accessed through the IConfig interface. The Monitor is the component responsible for capturing a set of metrics and providing them to the VNManager. It has two interfaces,

which are: IMonitorMetrics and INodeMetrics. The metrics captured are (i) specific metrics of the VN container (e.g., free memory, number of processors used, threads running, total of threads, peak of threads), obtained by using the INodeMetrics interface; (ii) physical metrics of the edge node that hosts the container (e.g., free physical memory size, the total physical memory size and the number of available processors), and (iii) network latency to reach this node, calculated through a ping operation.

The following components are deployed in the Cloud Tier: VNServiceDeliveryMgr (VNSDM); SysManager (SM); and ResourceAllocationMgr (RAM). The VNSDM and SM are entry points that receive requests issued the End-User and by applications via Infrastructure Provider (InP), respectively. The VNSDM is the same component described in the Edge tier but deployed at the Cloud tier to manage End-User requests that enter the system via the Cloud tier. The SM provides a set of Application Programming Interfaces (APIs) through the LightWeightManagementInterface (ILWM). It allows Infrastructure Providers (InPs) to manage the edge-cloud system and for instance, execute the registry operation of a VN by invoking the Registries Repository component using the IConfig interface. The RAM component deployed at the Cloud tier is the centralized resource allocation component in charge of engaging the master edge nodes in identifying the node (slave or master) capable of meeting the received application request.

3.2.1.1 Behavioral View

As mentioned in section 3.1.1, our architecture is designed based on a mix of microservice [11] [20] and container-based solutions [22][15]. According to [19], the containerization emerges as an approach that brings several benefits in an environment of high heterogeneity and resource-constraints, such as Edge computing. These benefits are related to the rapid instantiation, initialization and fast resizing of the resources without the overhead of restarting the system. Moreover, the use of containers facilitates the distribution and management of components on the edge nodes in contrast to other virtualization solutions such as the hypervisor [5]. In turn, the microservice is used to develop the framework components with a loosely coupled and highly cohesive design, thereby implementing a unique responsibility.

During the boot of our virtualization system, the components that encompass both the Edge tier (except the Virtual Node) and Cloud tier are loaded and initialized. It is worth to mention that the components of both tiers can be loaded independently with each other. Moreover, as our components are packaged in containers, we assume that each edge node already has the container images necessary to run the VNs. Therefore, we avoid incurring any network overhead, since there is no need of transferring container images between edge nodes.

Requests received at the Cloud tier are managed by the entry points to meet requests issued by applications via the Infrastructure Provider (InP) and the End-User respectively. However, each component has specific responsibilities. The VNSDM is the component in charge of managing the End-User requests. It offers a set of APIs through the IVNSD interface to allow users: (i) request data/events to the system, (ii) send an actuation command to the VN for execution, and (iii) discover registered VNs. The VNSDM goal is to receive those requests that arrive at the system (either at the Cloud tier (CT) or the Edge tier (ET)) and forward them to the ResourceAllocationMgr component. An implementation of the VNSD is also deployed at the Edge tier to provide an entry point for enabling the application requests arrival directly at the ET without going through the CT. The SM provides a set of Application Programming Interfaces (APIs) through the LightWeight Management Interface (ILWM). It allows Infrastructure Providers (InPs) to manage the edge-cloud system and for instance, execute the registry operation of a VN by invoking the Registries Repository component using the IConfig interface.

The centralized ResourceAllocationMgr (RAM), deployed at this tier. The RAM deployed at the Cloud tier is the centralized component in charge of engaging the master edge nodes in iidentifying the one node (slave or master) capable of meeting the received application request. Whenever a suitable edge node is identified, the RAM forwards the application request to it. Otherwise, the request is refused. Successful requests are then treated at the Edge tier by the local RAM, which offers the ResourceAllocatorInterface (IRA) used to receive the requests arriving via the VNSDM or forwarded by the centralized RAM. Upon receiving the application requests, the RAM invokes the VNInstanceMgr (VNIR) using the IVNIR interface to find a VN instance matching the request. When a VN instance that matches the requests is not found, or if the available VNs are busy (with other, previously received request), the RAM should make a decision regarding the current request. The decision should take into account the type of the edge note: (i) if it is a slave edge node, then the request is forwarded to its respective master edge node; (ii) if it is a master edge node, then the horizontal collaboration process is executed to find a neighbor master node capable of provisioning a VN.

Requests are then dealt by the ResourceProvisioningMgr (RPM). Initially, the RPM invokes the RegistriesRepositoryMgr (RR) using the

IConfig interface to seek a VN description that meets the application request. Then, the RPM executes the action to select and prepare the underlying physical infrastructure that is capable of hosting and running a VN instance according to the respective description. However, there are 3 exceptions that should be handled: (i) if a VN description is not found (so, the application is requesting a service not currently being provided by the edge-cloud infrastructure), or (ii) if a selected edge node becomes unreachable or (iii) if a selected edge node has not enough resources to host and running the VN, then the RPM is not able to proceed, so it sends a warning message in response to the application request.

Upon finalizing the above tasks with success, the RPM invokes the VNManager (VNM) component using the VirtualNode Interface (IVN) to instantiate the new VN. Initially, the VNM invokes the RR component through the IConfig interface to get the data type setting related to the request. Then, it identifies the type of VN and executes the VN instantiation. From now on the behavior depends on the type of VN. The VN exposes the IVN interface to provide its services upon receiving the requests from the ResourceAllocationMgr. Also, the VN operations are supported by engaging the Data Handler, SAM, and DSM. The interaction among these components is described as follows.

The VN of type actuation (VNac) invokes the SAM component using the IPUT interface to perform the requested actuation. The VN of type sensing (VNse) interacts with the DSM for retrieving the data streams from historical databases maintained at the Edge tier. The VNse can also invoke the SAM using the IGET interface whenever the data freshness of the stored date does not meet the target QoS requirement. In this case, fresh data must be acquired from the physical nodes at the Sensor tier. The VN of type data fusion (VNdf) (and its subtype VNcache) sends queries to the Data Handler (DH) component to fulfil its task. The DH, by its turn, queries the DSM, through the IEvent interface, to obtain the data streams needed to answer the VNdf request. The VN of type user function (UF) also interacts with the DSM for retrieving the data from historical databases. However, it performs user injected code (application specific functions) over data before returning the output data to the application. Finally, the VN of type event (VNevent) receives event data from the DSM and sends them to the application using a callback.

The behavior of the Sensor tier is centered around the SensingandActuationMgr (SAM) component, which starts, after the system booting, getting raw sensing data from the Sensor tier and send them to the DataStorageManager (DSM) to be stored.

4 THE IMPLEMENTATION ARCHITECTURE USING EDGEX FOUNDRY

In this Section, we present the proposal of architecture using Edgex Foundry. Figure 6 illustrates the architecture composed of specific components from the EdgeX Foundry framework [12] and third-party components. We used the EdgeX Foundry Components (EFC) for supporting the implementation of the components that encompass only our Edge tier (ET) since the Cloud tier (CT) has no support in EdgeX Foundry. The EdgeX Foundry is an open source framework designed for IoT Edge computing that encompasses a set of plug-and-play and loosely coupled microservice.

In LW-Dc4EC we are using the EFCs components to compose: (i) the Sensing and Actuation Manager (SAM) component tasks regarding the interaction with the physical environment, such as getting sensing data, performing actuation, and managing and communicating with the Sensor tier; (ii) Data Storage Manager (DSM) to manager the temporary database of the sensing data. Moreover, other EFCs are used to control the registry repository, export sensing data from the temporary database to cloud and clear the temporary database. The third-party components are used to (i) provide historical data processed or unprocessed in response to the application requests at the Cloud tier, (ii) a lightweight and high-performance message system, and (iii) a Complex Event Processing (CEP).

The EFC Device Services (DS) is the component composing the SAM in charge of receiving raw sensing data (after the boot of the system) from elements that encompass the Sensor tier and send them for storage in the temporary database using the EFC Core Data. Each DS is an edge connector in charge of abstracting the heterogeneity the Devices or IoT objects and allowing the interaction with them. These devices or IoT objects include, but are not limited to home appliances, alarm systems, heating and air conditioning, lighting, industrial machines, irrigation systems, drones, traffic signals, automated transportation, and so forth.

The edge connector is a component that encompasses the driver interface responsible for interaction with the physical device, service for data transformations, and handlers for servicing requests. Moreover, the DS interacts with the EFC Metadata to support its tasks. The SAM is also composed by EFC Command. It is responsible for providing the essentials interfaces to support the tasks of the Virtual Nodes (VN) to send actuation commands (interface IPUT) and get fresh data from the sensors directly (interface IGET). The EFC Metadata is an important component used to stores information about the services, devices and

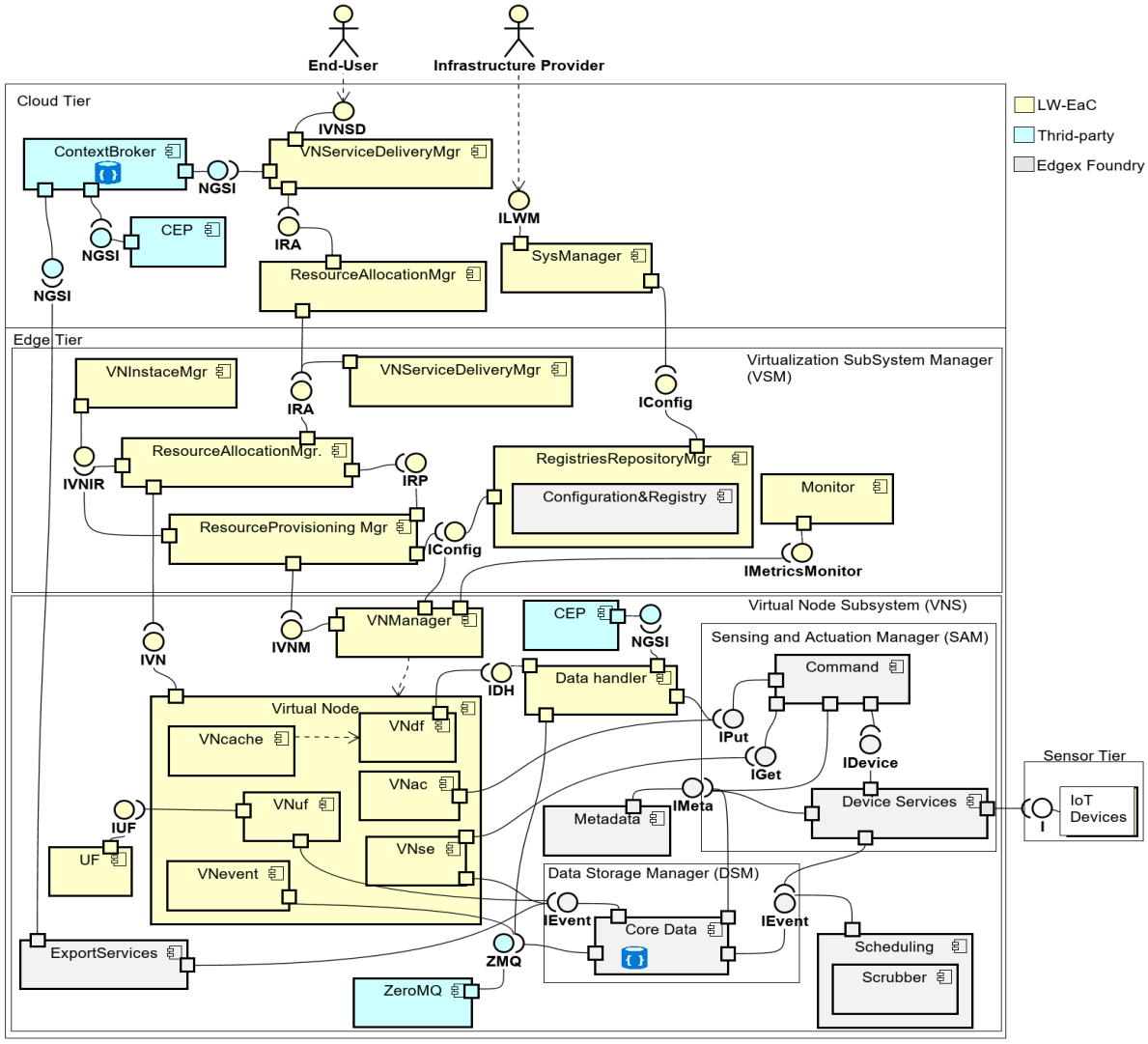

Figure 6: Implementation architecture

sensors (type, and organization of data) that are used by EFC Command, EFC Device Services, and EFC Core Data.

Regarding the cloud tier, we can observe both the LW-Dc4EC and third-party components. The third-party components encompass both the Context Broker and the Complex Event Processing (CEP) components. The Context Broker [21] is used as the provider of the historical data to the applications. The historical data are data that are no longer necessary to be available at the Edge tier. They are essential for applications that need to perform, for instance, a temporal analysis of the data. The Context Broker provides its services through NGSI interface [13]. The NGSI is a protocol in charge of providing a simple yet powerful open API that implements a RESTful API for Context Management. The CEP [8] is responsible for the execution of the queries over raw data from the Context Broker to provide data processed.

The EFC Configuration&Registry is the foundation to implement the Registries Repository (RR). It provides a database for persistence metadata besides the essentials APIs (store, remove and retrieve) to the management. The EFC Core Data is the foundation to implement the DSM. It is the component in charge of storing and retrieving the data stream from historical databases maintained at the Edge tier. The EFC Core Data provides a centralized persistence facility for data readings collected at the Sensor tier and uses a REST API for moving data into and out of the local storage. It also provides a degree of security and protection of the data collected while the data is at the edge.

The CEP at the Edge tier is the component in charge of executing queries over sensed data for the data fusion

VN type (VNdf) using the Data Handling (DH) component as intermediate. To provide the CEP with data, the DH obtains the data stream from the EFC Core Data through an asynchronous process (subscribing to a queue) using ZeroMQ. ZeroMQ [37] is a high-performance asynchronous messaging library, aimed to be used in distributed or concurrent applications. The VNS also includes the EFC Export Services, a component used to distribute data for other components or applications. In our architecture, it is used to move the data stored in the temporary database from the EFC Core Data to the Context Broker at the Cloud tier.

Finally, the EFC Scheduling is a component that can be used to schedule invocation of a URL. It includes the Scrubber microservice which cleans up the event and reading data that has already been exported to the Context Broker. Optionally, the Scrubber can also be configured to remove the stale event/reading data not exported.

5 RELATED WORK

Two recent works brought significant advances to the field of light virtualization models for sensors/IoT devices. Madria et al. [16] proposed a centralized virtualization model for Clouds of Sensors (Cos), which encompasses Virtual Sensors and provides sensing as a service for the users. Unlike Madria et al. [16], we implement a decentralized virtualization model tailored to meet requirements of emergent IoT applications such as low latency and location-awareness. In Santos et al. [25], the authors proposed Olympus, a decentralized and information fusion-based virtualization model for CoS. Olympus uses information fusion to ensure that the system will provide data at the abstraction level desired by each application (either raw or aggregated according to different levels from the feature to the decision level). In Santos [26], the authors extended the original design of Olympus to create a three-tier CoS infrastructure to provision Virtual Nodes (VN) at the edge of the network. Our proposal differs from Olympus and its extension in two essential aspects. First, we provide a process of collaboration between the VNs to actively share fresh data with neighboring VNs. Thus, we avoid re-reading the sensors to get the same data, thereby improving response time, bandwidth consumption, and sensor lifespan. Second, Olympus defines the VN as a program able to perform a set of information fusion techniques based on application requirements. Unlike Olympus, our model is more generic and provides predefined types of VNs representing each data type provided to serve the application requests.

Shi et al. [29] provide a flat view of Fog Computing that connects the cloud of sensors and smart devices via mobile devices. Their proposed infrastructure offers and consumes resources and services of mobile devices through the REST pattern using the CoAP protocol, thereby promoting the dissemination of data between users in a decentralized way. Unlike such work, we designed a new virtualization model capable of running VNs for providing sensing data or performing actuation in response to the application requests directly at the edge of the network. Moreover, we implemented a process of collaboration to allow VNs to share data with their neighboring nodes without the user mediation, thereby improving the request response time and saving bandwidth.

Concerning the collaboration between edge nodes, Taleb et al. [33] introduced the "Follow Me Edge". It is a concept based on Mobile Edge Computing (MEC) providing a two-tiered architecture to enable the migration of containers across edge nodes according to the localization of their mobile users. Although the container migration emerges as a feasible solution for the mobility requirement, the authors claim that the selection of the proper technique to perform the migration is a challenge to avoid both communication latency and data synchronism issues. Our proposal differs from Taleb et al. [33] by providing a collaboration process to share only the sensing data between edge nodes. Thus, we avoid transferring huge container images through the network, since each edge node already provides its services as VN containers thereby saving bandwidth and decreasing latency.

Wang et al. [35] present the Edge Node Resource Management (ENORM), a framework for handling the application requests and performing the workload offloading from the Cloud to running at the Edge network. ENORM addresses the resource management problem through a provisioning and deployment mechanism to integrate an edge node with a cloud server, and an auto-scaling tool to dynamically manage edge resources. Although our work was inspired by ENORM, our proposal is fully decentralized at the edge network. Such feature enables the edge nodes to find or provision the best VNs for providing either raw or aggregated sensing data, or performing actuation in response to the user application requests arriving from the cloud or the edge of the network.

Sahni et al. [24] present a novel computing approach named Edge Mesh integrating the best characteristics of the Cloud Computing, Edge Computing, and Cooperative Computing into a mesh network of edge devices to decentralize decision-making tasks. It enables collaboration between edge devices for data sharing and computation tasks. However, the authors present several open issues for implementing the communication between different types of devices. Some open issues concern how and which data should be shared between edge devices, and the appropriate local to execute the

intelligence of the application at the edge of the network. Our proposal leverages the advances promoted by the Edge mesh approach and addresses the related open issues by proposing all the steps for a collaboration process at the edge tier for enabling the data sharing between VNs.

6 FINAL REMARKS AND ONGOING WORK

The edge-cloud computing systems require a lightweight virtualization approach, to deal with the resource constraints of edge devices. Our proposal is adopting a virtualization approach based on containers and microservices, thus providing a virtualization model with low overhead. Moreover, we propose a data-centric approach, in which the virtual nodes are defined based on the data (either raw or processed), instead of on virtual machines or processing cores. Therefore, resources offered by the edge-cloud infrastructure, as well as application requests issued by end users, are described based on the data to be provided/consumed.

Our data-centric virtualization model leverages data reutilization among different application with similar requirements in terms of data sources, thus promoting higher return-of-investments for infrastructure providers. Our virtualization model provides several built-in types of virtual nodes that support the definition of different types of data-driven resources that are managed by the edge-cloud infrastructure. The definition of data-centric virtual nodes allows for various types of granularity in the content of a node, in order to promote either the reuse (sharing) of resources either the fulfillment of application-specific requirements. A virtual node can be specified to be tailored to the requirements of a single specific application, an application domain, or represent a generic function of data fusion or event detection. This feature helps dealing with the high heterogeneity of application requirements in edge-cloud systems.

The proposed software framework was specially designed to address the inherent challenges of resource management in edge-cloud ecosystems. The specified software components and description of their behavior will provide the underpinning and well-formed guidelines for building concrete resource management systems for these systems. Adopting a distributed, hierarchical approach to the framework and supporting collaboration between edge nodes enable addressing the challenges of large-scale, device heterogeneity, resource constraints, and also helps meeting application privacy requirements. Hierarchical approaches are well known for minimizing coordination overhead in large-scale systems, since master nodes are responsible for controlling their slave nodes, and message exchange is restricted to a smaller region, rather than requiring dissemination through all the nodes of the system. One can also take advantage of the heterogeneity of nodes, in order to assign the role of masters only to nodes with greater capacity of resources.

Regarding security requirements, the high availability of data produced by end users IoT devices raises privacy issues. For example, analyzing photos and videos generated by a smartphone can help identifying terrorist attacks or other public safety situations. Being able to have such data to be consumed by data analytics applications in the cloud can bring countless benefits not only to the device owner but to the community as a whole. Therefore, on the one hand, it would be important to share this data, but on the other hand, such information is often private/ confidential and cannot be disseminated blindly. The main challenge is to maintain user privacy while provisioning such analysis services. The proposed hierarchical approach can be extended to address this challenge. Each user can register her/his devices on an edge node in the vicinity, which would be considered her/his private edge node, and provide computing and storage capabilities. The raw data generated by the user would be associated with VMs instantiated on the private edge node, which could filter, preprocess and anonymize the relevant data before passing it to higher levels of the hierarchy for further analysis.

Our research team has concretized the concepts described in this paper into a functional prototype that is being currently tested within the scope of a Dell-funded R&D project. However, for confidentiality reasons, we cannot release the link for access to our prototype at this stage. We can say, in advance, that our functional prototype was implemented using the open source platform EdgeX Foundry [12].

ACKNOWLEDGEMENTS

This work is funded by Dell-EMC Brazil (grant number 0058) and by FAPESP (grant 2015/24144-7). Professors Flavia Delicato and Paulo Pires are CNPq Fellows.

REFERENCES

[1] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer Networks*, 54(15), pp.2787-2805, 2010.

[2] L. Bass, P. Clements, and R. Kazman, *Software architecture in practice*, Addison-Wesley Professional, 2003.

[3] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," In Proceedings of the first edition of the

MCC workshop on Mobile cloud computing, pp. 13-16. ACM 2012.

[4] M. Bouzeghoub, "A framework for analysis of data freshness," In Proceedings of the International Workshop on Information Quality in Information Systems, pp. 59-67, ACM, June 2004.

[5] T. C. Bressoud, and F. B. Schneider, "Hypervisor-based fault tolerance," *ACM Transactions on Computer Systems (TOCS)*, 14(1), pp.80-107, 1996.

[6] S. Chatterjee, and S. Misra, "Optimal composition of a virtual sensor for efficient virtualization within sensor-cloud," In Proceedings of IEEE International Conference on Communications (ICC), pp. 448-453, IEEE, June 2015.

[7] R. Dautov, S. Distefano, D. Bruneo, F. Longo, G. Merlino, and A. Puliafito, "Pushing intelligence to the edge with a stream processing architecture," In 2017 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), pp. 792-799, IEEE, June 2017.

[8] M. Dayarathna, and S. Perera, "Recent advancements in event processing," *ACM Computing Surveys (CSUR)*, 51(2), p.33, 2018.

[9] F. C. Delicato, P. F. Pires, and T. Batista, *Resource management for Internet of Things*, pp 1-116, Springer International Publishing, ISBN: 978-3-319-54247-8, 2017.

[10] F. C. Delicato, P. F. Pires, and T. Batista, "The resource management challenge in IoT," In *Resource management for Internet of Things*, pp. 7-18, Springer International Publishing, ISBN: 978-3-319-54247-8, 2017.

[11] N. Dragoni, S. Giallorenzo, A. L. Lafuente, M. Mazzara, F. Montesi, R. Mustafin, and L. Safina, "Microservices: Yesterday, today, and tomorrow," In *Present and Ulterior Software Engineering*, pp. 195-216, Springer, Cham, 2017.

[12] EdgeX Foundry. Available at: https://www. edgexfoundry.org. Last access: August 2019.

[13] FIWARE NGSI Open RESTful API Specification. Available at: http://forge.fiware. org/plugins/mediawiki/wiki/fiware/index.php/FI-WARE_NGSI_Open_RESTful_API_Specification. Last access: August 2019.

[14] M. Inaba, N. Katoh, and H. Imai, "Applications of weighted Voronoi diagrams and randomization to variance-based k-clustering," In Proceedings of the Tenth Annual Symposium on Computational Geometry, pp. 332-339, ACM, June 1994.

[15] B. I. Ismail, E. M. Goortani, M. B. Ab Karim, W. M. Tat, S. Setapa, J. Y. Luke, and O. H. Hoe, "Evaluation of docker as edge computing platform," In IEEE Conference on Open Systems (ICOS), pp. 130-135, IEEE, August 2015.

[16] S. Madria, V. Kumar, and R. Dalvi, "Sensor cloud: A cloud of virtual sensors," *IEEE Software*, 31(2), pp.70-77, 2013.

[17] R. Mahmud, R. Kotagiri, and R. Buyya, "Fog computing: A taxonomy, survey and future directions," In *Internet of everything*, pp. 103-130, Springer, Singapore, 2018.

[18] S. Misra, S. Chatterjee, and M. S. Obaidat, "On theoretical modeling of sensor cloud: A paradigm shift from wireless sensor network," *IEEE Systems Journal*, 11(2), pp.1084-1093, 2014.

[19] R. Morabito, V. Cozzolino, A. Y. Ding, N. Beijar, and J. Ott, " Consolidate IoT edge computing with lightweight virtualization," *IEEE Network*, 32(1), pp.102-111, 2018.

[20] S. Newman, *Building microservices: designing fine-grained systems*, O'Reilly Media, Inc., 2015.

[21] Orion Context Broker. Available at: https://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/Publish/Subscribe_Broker_Orion_Context_Broker_User_and_Programmers_Guide_R3. Last access: August 2019.

[22] C. Pahl, and B. Lee, "Containers and clusters for edge cloud architectures -- a technology review," In Proceedings of 3rd International Conference on Future Internet of Things and Cloud, pp. 379-386, IEEE, August 2015.

[23] C. Roy, A. Roy, and S. Misra, "DIVISOR: Dynamic virtual sensor formation for overlapping region in IoT-based sensor-cloud," In Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC), pp. 1-6, IEEE, April 2018.

[24] Y. Sahni, J. Cao, S. Zhang, and L. Yang, "Edge mesh: A new paradigm to enable distributed intelligence in Internet of Things," *IEEE Access*, vol. 5, pp.16441-16458, 2017.

[25] I. L. Santos, L. Pirmez, F. C. Delicato, S. U. Khan, and A. Y. Zomaya, "Olympus: The cloud of

sensors," *IEEE Cloud Computing*, 2(2), pp.48-56, 2015.

[26] I. L. Santos, L. Pirmez, F. C. Delicato, G. M. Oliveira, C. M. Farias, S. U. Khan, and Y. Y. Zomaya, " Zeus: A resource allocation algorithm for the cloud of sensors," *Future Generation Computer Systems*, vol. 92, pp.564-581, 2019.

[27] M. Satyanarayanan, V. Bahl, R. Caceres, and N. Davies, "The case for vm-based cloudlets in mobile computing," *IEEE Pervasive Computing*, 8(4), pp. 14-23, 2009.

[28] E. M. Schooler, D. Zage, J. Sedayao, H. Moustafa, A. Brown, and M. Ambrosin, "An architectural vision for a data-centric iot: Rethinking things, trust and clouds," In Proceedings of the IEEE 37th International Conference on Distributed Computing Systems (ICDCS), pp. 1717-1728, IEEE, June 2017.

[29] H. Shi, N. Chen, and R. Deters, "Combining mobile and fog computing: Using coap to link mobile device clouds with fog computing," In the IEEE International Conference on Data Science and Data Intensive Systems, pp. 564-571, IEEE, December 2015.

[30] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, 3(5), pp.637-646, 2016.

[31] O. Skarlat, M. Nardelli, S. Schulte, M. Borkowski, and P. Leitner, „Optimized IoT service placement in the fog," *Service Oriented Computing and Applications*, 11(4), pp.427-443, 2017.

[32] O. Skarlat, S. Schulte, M. Borkowski, M. and P. Leitner, „Resource provisioning for IoT services in the fog," In Proceedings of the IEEE 9th International Conference on Service-Oriented Computing and Applications (SOCA), pp. 32-39, IEEE, November 2016.

[33] T. Taleb, S. Dutta, A. Ksentini,, M. Iqbal, and H. Flinck, " Mobile edge computing potential in making cities smarter," *IEEE Communications Magazine*, 55(3), 2017.

[34] S. Tarkoma, *Publish/subscribe systems: design and principles*, John Wiley & Sons, 2012.

[35] N. Wang, B. Varghese, M. Matthaiou, and D. S. Nikolopoulos, "ENORM: A framework for edge node resource management," *IEEE Transactions on Services Computing*, pp. 1-1, IEEE, 2017.

[36] S. Yi, C. Li, and Q. Li, "A survey of fog computing: concepts, applications and issues," In Proceedings of the Workshop on Mobile Big Data pp. 37-42, ACM, June 2015.

[37] ZeroMQ. Available at: https://rfc.zeromq.org /spec:23/ZMTP/. Last access: August 2019.

AUTHOR BIOGRAPHIES

Igor L. Santos received his Doctorate degree in Informatics in 2017 from the Federal University of Rio de Janeiro. He is currently a lecturer at Federal Centre for Technological Education Celso Suckow da FonsecaHis research interests include Industry 4.0, Internet of Things, Wireless Sensor and Actuator Networks and Cloud of Sensors.

Flávia C. Delicato is an Associate Professor at the Federal University of Rio de Janeiro. Her primary research interests are IoT, WSN, middleware and Edge Computing. She has published 2 Books and over 160 papers.

Paulo F. Pires is an associate professor at UFRJ and leader of the UBICOMP laboratory. In the last few years his research efforts have focused on applying software engineering techniques in the context of system development for the Internet of Things. He has co-authored two books on the IoT theme: Middleware Solutions for the Internet of Things (2013) and Resource Management for Internet of Things (2017).

Marcelo P. Alves has an M.S. in informatics and is pursuing a doctoral degree in informatics, both at UFRJ. He is a lecturer at Uniabeu University Center and an independent IT consultant. His research interests include Internet, and distributed systems, model driven software engineering, IoT, and software architectures.

Ana Cristina Oliveira holds a degree in Systems Engineering (1992), a degree in Electronic Engineering (1993) and a master's degree in Theory of Controls and Statistics (1997), from the Department of Electrical Engineering at the Pontifical Catholic University of Rio de Janeiro, Brazil. She is currently a Senior Manager and a Data Scientist at Dell EMC. She has experience in the areas of Machine Learning, Big Data and Applied Statistics.

Tiago Salviano Calmon is a researcher in the area of data science at Dell-EMC and a PhD student at UFRJ, Brazil. He has experience in the field of Modeling and control of Management Systems, Distributed Systems and Optimization.