IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received June 6, 2020, accepted June 27, 2020, date of publication June 30, 2020, date of current version July 13, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3006037

On the Continuous Processing of Health Data in Edge-Fog-Cloud Computing by Using Micro/Nanoservice Composition

DANTE D. SÁNCHEZ-GALLEGOS[1], ALEJANDRO GALAVIZ-MOSQUEDA[2], J. L. GONZALEZ-COMPEAN[1], SALVADOR VILLARREAL-REYES[3], (Member, IEEE), ALDO E. PEREZ-RAMOS[2], DIANA CARRIZALES-ESPINOZA[1], AND JESUS CARRETERO[4], (Senior Member, IEEE)

[1]Cinvestav Tamaulipas, Victoria 87130, Mexico
[2]CICESE, Unidad Monterrey, Ensenada 66629, Mexico
[3]CICESE, Ensenada 22860, Mexico
[4]Computer Architecture and Technology Area (ARCOS), Department of Computer Science and Engineering, Universidad Carlos III de Madrid, 28991 Leganés, Spain

Corresponding author: J. L. Gonzalez-Compean (joseluis.gonzalez@cinvestav.mx)

This work was supported in part by the Council for Science and Technology of Mexico (CONACYT) through the Basic Scientific Research under Grant 2016-01-285276, and in part by the Project CABAHLA-CM: Convergencia Big data-Hpc: de los sensores a las Aplicaciones from Madrid Regional Government under Grant S2018/TCS-4423.

ABSTRACT The edge, the fog, the cloud, and even the end-user's devices play a key role in the management of the health sensitive content/data lifecycle. However, the creation and management of solutions including multiple applications executed by multiple users in multiple environments (edge, the fog, and the cloud) to process multiple health repositories that, at the same time, fulfilling non-functional requirements (NFRs) represents a complex challenge for health care organizations. This paper presents the design, development, and implementation of an architectural model to create, on-demand, edge-fog-cloud processing structures to continuously handle big health data and, at the same time, to execute services for fulfilling NFRs. In this model, constructive and modular *blocks*, implemented as microservices and nanoservices, are recursively interconnected to create edge-fog-cloud processing structures as infrastructure-agnostic services. Continuity schemes create dataflows through the blocks of edge-fog-cloud structures and enforce, in an implicit manner, the fulfillment of NFRs for data arriving and departing to/from each block of each edge-fog-cloud structure. To show the feasibility of this model, a prototype was built using this model, which was evaluated in a case study based on the processing of health data for supporting critical decision-making procedures in remote patient monitoring. This study considered scenarios where end-users and medical staff received insights discovered when processing electrocardiograms (ECGs) produced by sensors in wireless IoT devices as well as where physicians received patient records (spirometry studies, ECGs and tomography images) and warnings raised when online analyzing and identifying anomalies in the analyzed ECG data. A scenario where organizations manage multiple simultaneous each edge-fog-cloud structure for processing of health data and contents delivered to internal and external staff was also studied. The evaluation of these scenarios showed the feasibility of applying this model to the building of solutions interconnecting multiple services/applications managing big health data through different environments.

INDEX TERMS Big health data, edge-fog-cloud, the health-IoT processing, the Internet of Things, microservice architecture.

I. INTRODUCTION

Internet of Things (IoT) is considered by the industry and the research community as a key enabling technology to radically improve health care services and applications such as ambient assisted living (AAL) [1]–[3] and remote patient monitoring [4], [5]. The health-IoT devices currently represent 40% of the total IoT devices [6]–[9]. This technology is also becoming pivotal for users of these devices to acquire important data (i.e., vital signs, and activity level) and to make

The associate editor coordinating the review of this manuscript and approving it for publication was Noor Zaman.

decisions for improving their health care [10]–[13]. Moreover, users, health care professionals, and organizations can obtain insights for improving decision-making procedures by using and processing data extracted from IoT devices and also from other sources such as medical images, studies, historical documents from expedients, and health records [14], [15].

To achieve insights and useful information to support critical decision-making processes, different management tasks are performed during the health data lifecycle. In a typical health data lifecycle, the data are acquired, preserved, processed, and analyzed by using multiple applications deployed at different types of infrastructures [16]–[18]. Moreover, insights and information are shared with users, physicians, professionals, and health care organizations [1], [6].

In this context, the edge, the fog, the cloud, and even the end-user's devices play a key role in the management of the health sensitive content/data lifecycle [19]–[22]. This relevance is evident in scenarios where: the data are acquired from health devices and preprocessed at the edge; large volumes of data are processed at the fog; data are processed to produce useful information at the fog/cloud, or the information is visualized in end-users' devices.

For example, data are acquired from health-IoT devices [23]–[25], preprocessed at the edge [26], [27], processed at the fog/cloud to produce useful information [22], [28]–[31], and then this information is visualized in end-users' devices [32], [33]. Thus, the edge, the fog, the cloud, and even the end-users' devices play a key role in the lifecycle of the management of sensitive contents (i.e., healthcare data).

In real scenarios of processing IoT health data, the tasks executed at the different stages should be performed in either an automatic or semi-automatic manner to support critical decision-making processes [14], [34]. Also, the processing of health data requires to accomplish different non-functional requirements for attending health management norms and laws imposed by government and organizations [35], [36]. In this sense, the services for fulfilling non-functional requirements (NRF) such as security, reliability, and efficiency are also relevant for organizations to accomplish norms and regulations when managing health data.

The most relevant NFRs in real scenarios of management and processing of health data are, but not limited to, security, reliability, and efficiency. Security services are required to solve problems that arise when data and information are managed and shared with multiple users through non-controlled and untrusted environments. This is the case of the cloud (outsourcing models), where users lose physical control over the data [37]–[43]. In this context, data integrity, data confidentiality, and data access controls are important security aspects when using outsourced services (i.e., cloud).

The reliability services are required to solve problems related to outages in the infrastructure where the data are processed and stored as outages commonly result in the unavailability of the data [44]–[49]. This requirement results key for avoid end-users and organizations to suffer side-effects from data unavailability.

The efficiency is a key requirement to solve issues related to the costs of storage and transportation of the data through the different environments of processing [50]–[53], as well as to reduce delays in the delivering information required in decision-making processes, which are critical in real health scenarios [54], [55].

However, creating portable and flexible solutions that can be deployed on different environments (any combination of edge, fog, cloud, and/or end-users' devices) [56]–[59] by using multiple services and applications, and at the same time these solutions enforce the fulfilling of mandatory NFRs in continuous, transparent integrated and efficient manners is still an open research challenge [60]–[63].

Three main issues are involved in this challenge and are faced in this paper:

- The first issue is creating, in configuration and deployment times, solutions including multiple applications (functional components) that are executed by multiple users in multiple environments.
  In this paper, to face this issue is proposed an architectural model based on recursive maps of abstractions called *blocks* and execution environments (edge, fog, cloud, or end-user's devices). The goal of this model is to enable organizations to build edge-fog-cloud processing structures.
- The second issue is to produce dataflows through the *blocks* considered in an edge-fog-cloud processing structure.
  In this paper, this issue is solved by using continuous delivery schemes, which establishes controls on the execution sequence of the *blocks* in the defined environments as well as establishing channels for the data exchange between pairs of *blocks*. This scheme produces uninterrupted dataflows from the IoT devices through the *blocks* deployed on any of the edge, fog, cloud, or end-users' devices to the end-users of the solution.
- The third issue is the fulfillment of NFRs for each data arriving/departing to/from each *block* in an edge-fog-cloud processing structure.
  In this paper, this issue is solved by continuity schemes of NFRs, that are coupled to the *blocks* to ensure the fulfillment of NFRs for each data arriving/departing to/from each *block* in an automatic and transparent manners. These schemes produce a continuous fulfilling of NFRs through the different environments where a solution has been deployed on (any combination of edge, fog, cloud, or end-users' devices).

This paper thus presents an architectural model that enables organizations to build processing structures deployable on any combination of edge, fog, or cloud environments for health decision-making processes. This architectural model creates *edge-fog-cloud processing structures* by using maps based on recursive and modular abstractions called *blocks*.

A *block* is built as a self-contained reusable software piece that includes an application (for processing health data) as well as services for enforcing the fulfilling of NFRs and I/O interfaces for coupling it to other *blocks*. In this model, depending on the deployment environment, the *blocks* are classified in regular (for the cloud and the fog) and *microblocks* (for the edge). A *block* is managed by an execution map that considers: *i)* the place where a *block* will extract data from any source during execution time, *ii)* the environment where that *block* will be executed in (i.e., edge, fog or cloud), and *iii)* the place where the data transformed by that *block* will be stored in. An *edge-fog-cloud processing structure* thus is created by coupling the I/O interfaces of a set of *blocks* (by using the execution maps of the *blocks*). The *blocks* are implemented in the form of nanoservices[1] or microservices[2] [66]–[68], which are abstract representations of modular and independent software pieces.

A *continuous delivery* scheme (*CD*) is a map that is used in this model to perform, in automatic and transparent manners, the correct deployment and execution of the *blocks* (by using execution maps), as well as the continuous data delivery to the *blocks* of an edge-fog-cloud structure, which will be deployed on any combination of edge, fog, cloud, or end-user devices. The *continuity schemes* enforce the fulfilling of non-functional requirements (NFRs) in the management of the data arriving and departing to/from the *blocks* encapsulated into an edge-fog-cloud processing structure. These continuity schemes are created by taking advantage of the recursive creation and management of *blocks* proposed in the architectural model. This means that the continuity schemes are created by using *reserved blocks*. This type of *block* is coupled to the *blocks* of an edge-fog-cloud processing structure to create continuous and simultaneous enforcement of the fulfilling of the NFRs associated with each functional *block* of an edge-fog-cloud processing structure.

The following continuity schemes were created for the architectural model presented in this paper:

- Continuous security (*CS*), for producing integrity, confidentiality, and access controls at each stage of the sensitive data processing lifecycle. The reserved *blocks* considered in this scheme are managed by maps linking a *block* with integrity, confidentiality, access control, and signatures services.
- Continuous reliability (*CR*), for adding fault-tolerance to data for withstanding lost data as well as cloud outages and/or fog servers unavailability. Reliability is managed as a map linking a fault-tolerant service with a *block*, whereas the *blocks* are recursive as the different services for fulfilling NFR are also created by using *blocks* and *microblocks*.
- Continuous efficiency (*CE*), for improving the delivery and processing of data by applying implicit parallelism on the processing tasks, and data compressing to reduce the volume of the data stored. A map for linking software pieces that creates parallel patterns with a *block* of an edge-fog-cloud solution is considered in this scheme. These pieces are also managed as reserved *blocks*.

The continuity schemes thus are managed as a matrix of NFRs (see security, reliability, and efficiency requirements in Fig. 1) and deployment environments (see edge, fog, cloud, and end-users' devices environments in Fig. 1).

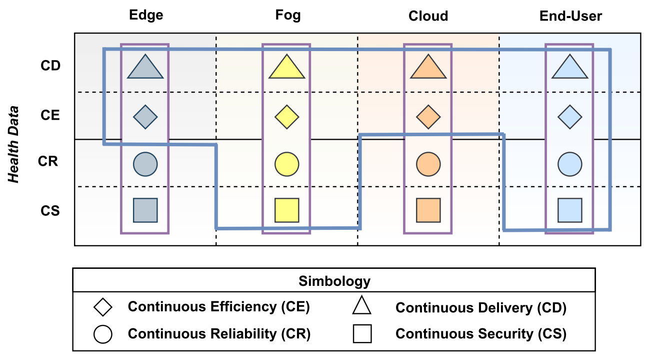


FIGURE 1. Conceptual representation of a matrix of non-functional requirements and deployment environments managed by the continuity scheme maps of the architectural model.

A conceptual representation of these continuity scheme maps applied to the *block* structures is depicted in Fig. 1. The area delimited with a blue line represents an example of an edge-fog-cloud-EndUser processing structure, including four *blocks* (one per environment). The set of continuity schemes that will be included in each *block* can be defined as required. In this example, all *blocks* include $CD + CE$ (automatic data delivery and continuous efficiency), whereas the *blocks* at the fog and end-users' devices also include $CD + CS + CR$ schemes.

The basic idea of the proposed architectural model is that the developers of the organizations can create comprehensive solutions in three simple steps:

1) Developers associate an either analytic or processing application to a *block*, which, in an implicit form, will be in charge of the management of data I/O operations.
2) Organizations create edge-fog-cloud processing structures by defining the sequence in which *blocks* will be deployed on the different environments (the edge, the fog, the cloud, and the end-user devices). The deployment and coupling of *blocks* through different environments is also performed in implicit and automatic manners.
3) The organization chooses the continuity scheme for each *block* by enabling as many continuity schemes as needed to fulfilling requirements of laws/norms or enough for addressing users' concerns about the

[1]A nanoservice is a small software piece created by using a template invoking an application, I/O calls, and a reduced configuration file, which is suitable for edge environments.

[2]A microservice, in this architectural model can include applications, database, and a set of I/O interfaces. This type of piece is encapsulated into a portable light virtual container [64], [65] enabling the solutions to be deployed on any of the fog, cloud or end-users' devices.

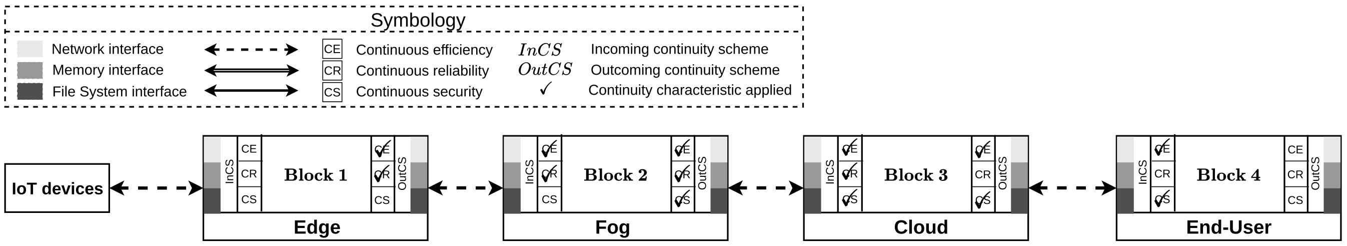


FIGURE 2. **Example of a an edge-fog-cloud processing structure managed by using microservices and nanoservices.**

environment in which the data will be processed in (see an example in Fig. 1).

Fig. 2 shows an example of *blocks* interconnected by input and output interfaces creating an edge-fog-cloud processing structure. This example also shows the continuity schemes for the management of NFRs of data arriving at the *block* and data departing from the *block* (see a checkmark for secure/*CS*, reliable/*CR*, or efficient/*CE* for each *block*). Notice that in Fig. 2 these *blocks* can be deployed in any combination of infrastructure environments to create edge-fog-cloud-EndUser processing structures to manage and process big health data by using maps of *blocks* and environments.

A prototype based on the architectural model proposed in this paper was built to create edge-fog-cloud structures. These structures were evaluated when extracting health data from the IoT devices, preprocessing them at the edge, processing resultant data at the fog, preserving resultant information at the cloud, and visualizing the results at the end-users' devices.

A case study was conducted based on these structures by considering scenarios where: *i)* end-users receiving insights discovered when processing electrocardiogram (ECG) data produced by IoT sensors; *ii)* physicians receiving warnings created from the analysis and identification of peaks in the ECG data based on remote patient monitoring produced by a wireless network of IoT devices deployed on an emergency room. The performance of the structures is compared with a state-of-the-art solution. *iii)* Simultaneous management of spirometry, medical imagery, and ECGs to support making-decision procedures; *iv)* organizations managing multiple processing structures including the management of ECG data, medical imagery, and spirometry. In addition, an evaluation study was conducted to the assessment of the impact of edge-fog-cloud processing structures on the storage utilization and the computation complexity of big health data solutions built by the model proposed in this paper.

The experimental evaluation revealed the feasibility of applying this model to the building of solutions interconnecting multiple services and applications for the management of big and heterogeneous sensitive data through different environments.

The main contribution of this paper includes are:

1) An architectural service composition model based on modular infrastructure-agnostic structures for edge-fog-cloud computing.
2) A multi-continuity model for the application of services fulfilling non-functional requirements that are critical for the management of sensitive information (health data and contents). The following schemes were developed and evaluated in this paper:
   - *Continuous security* schemes for establishing access controls, data integrity verification, and data confidentiality when structures *edge* ∨ *fog* ∨ *cloud* ∨ *EndUser* exchange sensitive data (i.e., medical imagery, ECG data, or spirometry data).
   - *Continuous reliability* schemes for adding fault-tolerance to data when structures *edge* ∨ *fog* ∨ *cloud* ∨ *EndUser* exchange and store sensitive data.
   - *Continuous efficiency* schemes to convert applications/services used by structures *edge* ∨ *fog* ∨ *cloud* ∨ *EndUser* into parallel patterns improving not only the performance of these structures but also the cost-efficiency of data storage and transportation.
3) An implicit management scheme to control the continuous security, reliability, and efficiency schemes for the management of large volumes of data.

The rest of the paper is organized as follows. Section II describes the related work of this paper. Section III presents the design and implementation of the workflow architectural model based on microservices and nanoservices composition for edge-fog-cloud computing. Section IV presents the case study based on an emergency room, and also shows the results of the experimental evaluation conducted. Finally, Section V gives conclusions remarks and future work.

II. RELATED WORK

The challenge of the processing big IoT sensitive health data has been addressed in different scopes: isolated analytic applications, big data solutions, and some solutions focused on non-functional requirements (NFRs).

On the data analytic scope, different solutions have been proposed to extract relevant features from IoT devices [17], [69], [70]. Artificial intelligent (AI) algorithms [14], [71], [72] performing periodic monitoring over historical data have been proposed to discover patterns from the data collected. Moreover, applications and models are available for sending warnings to patients and physicians [73]–[75]. In this context, Zhang *et al.* [76], Chze and Leong [77] proposed solutions where different sensors and applications interact through multiple hops in continuous integration to create a dataflow between devices and decision-makers.

On the scope of big data scenarios, cloud-based solutions have emerged as the election for many health organizations to manage, preserve, and share large volumes of data in a cost-effective manner [78]. Different solutions have been proposed in the literature for managing, processing, and preserving health data [22], [79] by using big data tools like Hadoop [80], [81], and Spark [82].

Nevertheless, this type of solution could result either in the loss of control over the data sent to the cloud or in a vendor lock-in dependency, when the management of the accumulating data in outsourced processing services [37], [83], [84]. These issues may not be acceptable in health IoT data scenarios where it is crucial to guarantee the privacy, management, security, and integrity of the data [38]. In turn, the architectural model proposed in this paper avoids vendor lock-in scenarios and recover control of data when the cloud services are used to manage the IoT health data by using: *i)* portable infrastructure-agnostic structures to deploy applications/services (*blocks*), and *ii)* controls over the processing and management of data through continuity schemes fulfilling non-functional requirements in automatic and transparent manners.

On the scope of the management of NFRs the main proposals have been focused on security and reliability requirements. End-to-end processing schemes have been proposed to mitigate the lack of control over the data in outsourced environments [85]–[87]. In this type of scheme, data are preprocessed before sending them to the cloud by applying techniques such as encryption to reduce security issues [39], [88], [89], and reliability schemes based on coding and redundancy to face outages [47]–[49], [90] or failures in the cloud [44], [91]. Indexing and hashing techniques have been proposed to verify the integrity of the data [92], [93]. For instance, Moosavi *et al.* [94] proposed an end-to-end scheme to ensure the data before sending them to the transportation stage in a mobility environment. Authentication and authorization of entities participating are established in this scheme. SecFilter [95] is a security filter that applies different security policies to data in information-sharing environments before sending them to the cloud. The policies are automatically defined by the risk level that is discovered by SecFilter by using mining data techniques. Nevertheless, these solutions are focused on a specific environment such as the edge [21], [96]–[98], the fog [20], [99], [100], or the cloud [28], [101], [102]. Moreover, these solutions are only focused on the accomplishment of a specific NFR (security [103], [104], reliability [86], [105], or integrity [93], [106], [107]). The model presented in this paper differs from end-to-end solutions as the security and reliability requirements are fulfilling in implicit and automatic manners. These processes are independent of the environment where the applications are deployed.

The issue of deploying services and applications on different environments has been addressed by traditional computational workflows [86], [108]–[111]. However, in practice, users have to perform troubleshooting procedures to deploy a workflow on a given infrastructure [112]. Moreover, the heterogeneity of the stages in a workflow [113] and its schedule of processes [114] could produce an impact on the performance of the workflow in execution time.

Microservice architectures [66], [115] represents a solution for developers to convert large services into small, independent heterogeneous and isolated services that are managed by using exchange data and messages [116]. Nevertheless, the creation of comprehensive solutions with implicit resource management and flexible portability is still an issue for microservice architectures [117]. In turn, the model proposed in this paper introduces software pieces called *blocks* (implemented in the form of microservices and nanoservices), which include methods for managing the exchange of data and messages in implicit, secure, reliable, and cost-efficiency manners. Moreover, these *blocks* are managed by using recursive, reusable, and chainable structures encapsulated into virtual containers. These structures are self-contained, which means no troubleshooting is required and that such structures can be deployed on any of the edge, fog, or cloud. These structures allow users to create different processing structures that can be deployed on any combination of edge, fog, or cloud. Also, continuity schemes [85] simultaneously enforce the continuous delivery of data in each stage of a processing structure and to fulfilling the NFRs for data processed by the *blocks* of the processing structures.

A. QUALITATIVE COMPARISON OF CONTINUITY TOOLS

This section presents a qualitative analysis of the architectural model proposed in this paper and solutions identified in the state-of-the-art focused on the continuous processing of IoT data.

Table 1 shows a qualitative comparison between the proposed model, based on *blocks* and *microblocks*, and different solutions from the state-of-the-art focused on building workflows and pipelines. Table 1 includes traditional workflow engines for the building of processing solutions (i.e., Comps [118], Pycomps [119], Sacbe [86], Parsl [121], and DagOnStar [124]), and software for building distributed processing IoT dataflows based on message exchange (i.e., Apache Kafka [122] and Amazon kinesis [123]). The qualitative comparison was performed considering reliability, security, and efficiency features and the different applications of these non-functional requirements.

The reliability property was evaluated considering the data transfer point-of-view, data storage, and data processing. Data transfer reliability was assessed as the capacity of the solutions to provide data with fault tolerance in distributed infrastructures. Reliability in data storage refers to the capacity of the solutions to store data for withstanding failures in storage nodes, especially in the fog and in the cloud. Reliability in data processing refers to support fault-tolerant computing

TABLE 1. Qualitative comparison between the model based on *blocks* and *microblocks* with different workflow engines and IoT processing pipelines.

Work	IoT oriented	Reliability			Efficiency		Security		
		Transport	Storage	Processing (fault-tolerance)	Parallelism	Compressing	Integrity	Confidentiality	Access control
Sacbe [86]			✓	✓	✓				
Comps [118]				✓	✓				
PyComps [119]				✓	✓				
Makeflow [120]				✓	✓				
Parsl [121]				✓	✓				
Apache Kafka [122]	✓	✓		✓	✓	✓		✓	✓
Amazon kinesis [123]	✓	✓		✓	✓	✓			✓
DagOnStar [124]	✓			✓	✓				✓
Blocks and Microblocks	✓	✓	✓	✓	✓	✓	✓	✓	✓

mechanisms. As shown in Table 1, existing solutions only provide reliability in data processing, whereas the *blocks* and *microblocks* of the proposed model offer the three types of reliability evaluated in this analysis.

Efficiency was evaluated in terms of performance by considering whether a solution supports *parallelism* or not. In this sense, solutions like Parsl, Comps, Makeflow and DagOnStar produces implicit parallelism by using a multithreading model, whereas Apache Kafka and Amazon Kinesis provide it by manually cloning instances of the application and implementing load balancing services to distribute the load between the cloned instances. In the case of the proposed architectural model, the *blocks* produce implicit parallelism by using parallel patterns based on virtual containers. These patterns automatically clone *microblocks* (n processing workers) and implement not only task parallelism as do the evaluated solutions but also data parallelism, pipe&filters, shared resources, and combinations of these patterns. Moreover, the *blocks* also consider automatic load balancing and workload distribution. Furthermore, the *blocks* are self-contained and portable pieces of software, which produces infrastructure-agnostic solutions.

Three characteristics of security were considered: integrity, confidentiality, and access control. Integrity refers to the ability of the solutions to use checksums (e.g, SHA3 hashing or MD5) to ensure that data have not been modified in the transmission from one environment to another one [125]. Confidentiality was evaluated as the capacity of the solutions to preserve data privacy between processing stages and nodes [126], [127]. Finally, access controls were evaluated by considering techniques to establish cryptography-based controls to access the processing solutions [128].

Based on the previous discussion, it can be stated that the architectural model proposed in this paper has two main advantages. Firstly, it offers more quality features than available models, and secondly provides the solutions created by using this model with a continuous enforcing of offered features through the different environments where the solutions are deployed on. Thus, the proposed architectural model represents a quite useful tool for an organization to process big sensitive IoT health data.

III. A WORKFLOW ARCHITECTURAL MODEL FOR EDGE-FOG-CLOUD COMPUTING BASED ON MICRO/NANOSERVICES COMPOSITION

This section presents an architectural model based on the composition of *blocks* implemented as micro/nanoservices to build processing structures in edge-fog-cloud computing environments. This section also presents continuity schemes for simultaneously managing the exchange of data and messages through the *blocks* included in the edge-fog-cloud processing structures and to fulfilling, in a continuous manner, non-functional requirements (security, reliability, and efficiency) of data arriving and departing to/from the *blocks* considered in a processing structure.

A. AN ARCHITECTURAL MODEL BASED ON MICRO/NANOSERVICES COMPOSITION

This model considers a service composition based on an abstract representation of *nanoservices* and *microservices* called *blocks*. A *block* represents thus a self-contained reusable software piece that includes an application (for processing health data), I/O interfaces for coupling it to other *blocks* as well as services for enforcing the fulfillment of NFRs. A *block* is managed by an execution map that considers: *i)* the place where a *block* will extract data from any source during execution time, *ii)* the environment where that *block* will be executed in (i.e., edge, fog or cloud), and *iii)* the place where the data transformed by that *block* will be stored in.

A *block* can be chained to other *blocks* by using its I/O interfaces, for creating processing structures such as parallel patterns, pipelines, and workflows (see an example of pipeline structure including *blocks* in Fig. 2). The input and output interfaces of a constructive *block* are denoted as $In-I$ and $Out-I$ respectively, and they could be any of the memory (denoted by $\Longleftrightarrow$), the network (denoted by $\longleftarrow\!\rightarrow$) or the file system (denoted by $\longleftrightarrow$).

In this model, the *blocks* are classified into two types depending on the environment where the *blocks* are deployed on: the first one is a regular *block* that is implemented as microservices encapsulated into a portable light virtual container [64], [65]. This type of *block* can be allocated in any of the fog, cloud, or end-users' devices [66]–[68]. The second one is a small *block* or *microblocks* that is implemented

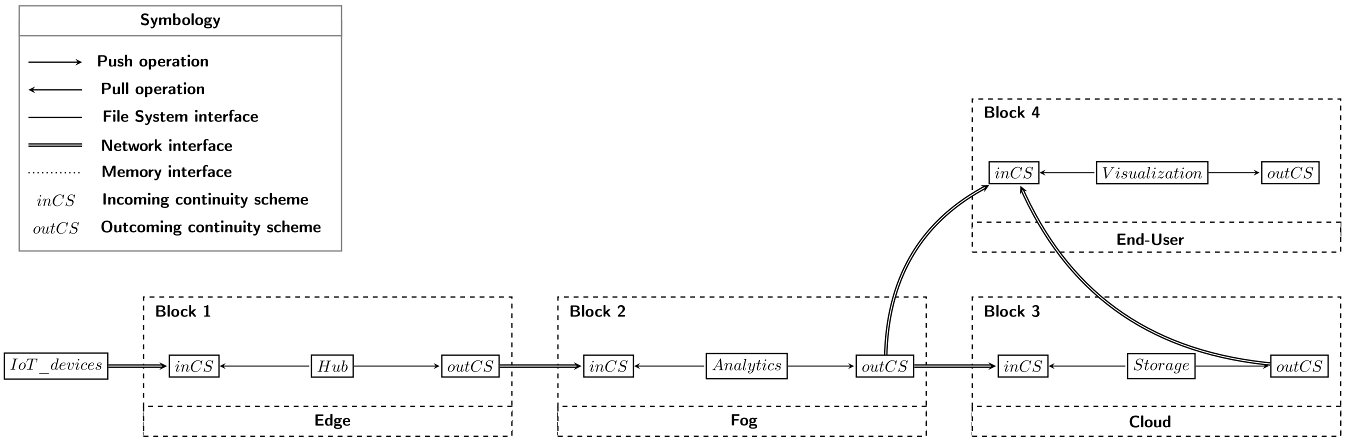


FIGURE 3. Example of *blocks* chained to create health workflows.

in the form of nanoservices, which are suitable for edge environments. See an example of *blocks* deployed in edge-fog-cloud-EndUsers environments in Fig. 2.

In this model, a microservice is denoted as $microS_x$, whereas a nanoservice is represented by $nanoS_x$. These structures are encapsulated into portable and light virtual containers (VC) [64] for avoiding troubleshooting or missing software dependencies as well as vendor lock-in dependency. As a result, a *block* mapped with a microservice can be deployed on the following environments (Env): the fog (fog), the cloud (Cl), and end-user' devices (EU). Moreover, a *block* mapped with a $microS_x$ could include a set of applications ($Apps$) or nanoservices and exposed them by an API that developers can consume with their apps, thus enabling the interaction with other $microS_x$. Therefore, a $microS_x$ is denoted as follows:

$$microS_x = \text{VC}(In - I, Apps \vee nanoS, Out - I, API). \quad (1)$$

A nanoservice, in turn, is a small software piece created by using a template invoking an application, I/O calls, and a reduced configuration file, which is suitable for edge (Ed) environments where the computational resources are limited. A *block* mapped with a $nanoS_x$ is built by using a template containing a function that can be used to process data, as well as the input and output interfaces ($In - I$ and $Out - I$ respectively). Thus, it is denoted as follows:

$$nanoS_x = \text{template}(In - I, function, Out - I) \quad (2)$$

The notation of $nanoS_x$ and $microS_x$ generalizes the definition of a *block* (Blk) and the control and management of data and workload in these structures are implicitly performed within the *blocks*.

1) BUILDING EDGE-FOG-CLOUD PROCESSING STRUCTURES FOR HEALTH DATA

This subsection describes a model for the construction of edge-fog-cloud processing structures. To build these structures, the first step is to define the *execution maps* which, as previously described, manage the *blocks* in execution time by using the notation of a nanoservice ($nanoS_x$) or a microservice ($microS_x$). These maps are built by following the ETL processing model traditionally used in big data applications: *i)* to extract data from a data source, *ii)* to transform the acquired data into information, and *iii)* to load the information to a data sink [129]. These maps convert a *block* into independent software that can be used in an isolated manner.

The second step is to define the *environment maps* for each *block* in a solution. These maps are created by linking a *block* (Blk) with either a nanoservice ($nanoS_x$) or a microservice ($microS_x$), which is associated to an environment ($Env_x = Ed \vee fog \vee Cl \vee EU$). A map ($map_x(Env, Blk)$) represents the association of a *block* (Blk) with a specific environment (Env) where the Blk will be deployed. Those maps are used to deploy the solutions on an edge-fog-cloud computing environment. In this architectural model, each map (map_x) produces a specific configuration file for each Env specified in that map.

The last step is to create the *coupling maps* by interconnecting the $Out - I$ of a *block* with the $In - I$ of another one. Thus, an edge-fog-cloud processing structure is created in configuration time by deploying *blocks* using a set of chained coupling maps. Fig. 3 shows an example of *blocks* organized as an edge-fog-cloud structure built, for example, to manage and process health data, and delivering the results to the end-users. In this example, *blocks* are chained by using the network interfaces.

At this point, the *blocks* can be related to functional aspects of a health processing (any of data analytics, visualization of information, or sharing of contents).

2) MAPS OF PARALLEL PATTERNS TO IMPROVE BLOCKS EFFICIENCY

The interconnection of *blocks* through different I/O interfaces defined in the maps produces patterns of *blocks* that create structures such as pipelines, workflows, and different parallel schemes. These structures are focused on improving the functional properties of end-fog-cloud solutions. The pipelines and workflows enable developers to create comprehensive solutions in one single environment. For instance, an analytic procedure could include a pipeline of three *blocks*: one *block* for data preparation, the second one for classification, and the third one for indexing resultant information. The same example can be applied to a workflow structure but deploying the three *blocks* on different environments (i.e., the first one on the edge, the second one on the fog, and the last one on the cloud). In (3) is showed the notation to build this workflow structure with tree *blocks* (blk_1, blk_2 and blk_3) deployed at the edge, fog, and cloud environments.

$$PattBlks = map_1(edge, blk_1) \rightarrow map_1(fog, blk_2) \rightarrow map_1(cloud, blk_3) \quad (3)$$

In turn, the parallel patterns are managed to improve the efficiency of *blocks*. These patterns consider to clone a given *Blk* and organize them in the form of a task-parallel or data-parallel method. A parallel pattern could be represented by the following expression:

$$Blk_{patt} = (src, pattern, Blk, cl, snk), \quad (4)$$

where *src* represents the data source, *pattern* represents the type of parallelism, *Blk* is the *block* to be executed in parallel, *snk* the data sink, and *cl* the number of *Blk* clones. In practice, Blk_{patt} represents a parallel version of *Blk* as both perform the same functionality. We recall that a *Blk* can be implemented as either microservice or nanoservice encapsulated into a virtual container. This produces a recursive scheme where *Blk* represents a *microblock* for Blk_{patt}.

As a result of this coupling, *cl* replicas of the application linked with *Blk* are executed in a concurrent manner processing data/workload in parallel. Notice that *snk* of the Blk_{patt} will be linked to either a *Blk* output or a data Sink. This means this pattern can be chained to other patterns (any of pipeline, workflow, or another parallel structure) in a recursive manner and the map scheme of this model supports this recursive feature.

In this paper, we have considered two types of patterns: *i) mw* for *manager/worker* patterns used to process a large volume of data (i.e., digital electrocardiogram produced constantly); and *ii) dq*, created by using the *divide&conquer* technique to process files of large size (i.e., medical images and spirometry studies).

A *manager/worker* (*pattern* = *mw*) pattern considers reserved *blocks* associated to two software artifacts called *manager* and *workers*. The *manager* performs the following actions: *i)* the creation of the *Blk* clones (*workers*); *ii)* the interception of the data arriving at the *block*; *iii)* the creation of *tasks* (where a task is equal to a content, a file, or a record in a database) of workload to be processed; *iv)* the distribution of these *tasks* to the *workers* by using a load-balancing technique; and *v)* the supervision that each worker collects *tasks*, produces results, and delivers them to the output of the sink (i.e., a folder, or the input interface of the next *Blk* or pattern) [130], [131]. The *workers* are in charge of two functions: *i)* to execute the applications/services associated to the *Blk* to process the tasks sent by the *manager*, and *ii)* to deliver their results to an $In - I$ of a *block* or a *sink*.

The second technique used to create parallel patterns is *divide&conquer*. This technique is focused on contents of big size (i.e., tomography images) and considers three *blocks* (artifacts): the *divide*, the *workers*, and the *conquer*. The *divide* is in charge of splitting the input data into segments, which are delivered to the *workers*. The *workers* process the segments and deliver results to the *conquer* entity. And, the *conquer* consolidates the results into a single one and delivers consolidated results to either an $In - I$ of a *block*, or a *sink*.

3) CONTINUOUS DELIVERY SCHEMES BASED ON COUPLING MAPS

At this point has been described the process of building edge-fog-cloud processing structures in configuration and deployment times. In this section, is described the method for delivering data through all the components of an edge-fog-cloud processing structure in a continuous manner.

Push and pull methods [132] were designed to establish controls over the execution of *blocks* and the data exchange through the *blocks* of an edge-fog-cloud processing structure. These methods produce a dataflow by following coupling maps of the solution. These methods are applied to each pair of *blocks* in a solution. Push means that the first *block* is in charge to send the data to the second *block*, whereas pull means that the second one is in charge to go to the output interface of the first *block*. This means that when using a single method to set up a block-to-block communication, an asynchronous channel is established between the pairs of *blocks*. In turn, a synchronous channel is established where both *blocks* are using both modes (one using push and the other one is using pull). In any case, the channels produce a continuous exchange of data (dataflow). These operational modes, for instance, are used when an edge node sends data to the fog (push mode), the data are processed and the results are sent to the cloud (push mode), and an end-user's device requesting insights of the data from the cloud (pull mode).

A *continuous delivery* scheme is built when configuring these modes for following the coupling schemes of an edge-fog-cloud processing structure. To add these dataflow methods to the model, the push mode has been denoted as a right arrow from the *block* that is sending the data to the *block* that is receiving the data, such as follows: $map_1 \rightarrow map_2$. Whereas, the pull mode is denoted as $map_1 \leftarrow map_2$, which means that the map_1 is requesting data from the map_2.

These operation modes are used in run-time for controlling data exchange in a decentralized manner to avoid dependencies among *blocks*. A bidirectional operation is required for delivering the status of the push and pull operations to the *blocks* involved in these operations (synchronous communication). This is represented as $map_1 \leftrightarrow map_2$.

Following this notation, a workflow composed of n coupling maps would be denoted by the following expression:

$$Wf_x = map_1 \longleftrightarrow map_2 \longleftrightarrow \ldots \longleftrightarrow map_n \quad (5)$$

In this example, the behavior of a workflow (Wf_x) in run-time is similar to a *block*: *i)* the workflow extracts data from a source (src_x), which could be any health-IoT device (for example a spirometer, electrocardiogram, or a health wearable) or a static data repository; *ii)* the workflow transforms the acquired data into either useful information or a new version of the acquired data by executing the considered *blocks*; and *iii)* the workflow loads the results in a sink (snk_x) or another *block* (i.e., in an end-user's device).

In this sense, a health workflow [133] could be represented as a map of a data source (src_x), a processing workflow (Wf_w), and a data sink (snk_x). This is denoted as:

$$HWf_x = (src_x, Wf_x, snk_x) \quad (6)$$

To clarify the usage of this notation, an example of an edge-fog-cloud-EndUser processing structure built by using this notation is showed in (7). This processing structure was built using five maps (one per processing stage). Each map includes two metadata structures to define the environment (Env) and a *block* (blk). Maps are interconnected with other maps through an input/output interface ($In - I$ and $Out - I$). Therefore, the first two maps are interconnected through a memory interface (mem), whereas the rest of the maps are interconnected through a network interface (net).

The *blocks* associated with the first and second maps indicate the deployment of *blocks* on a personal computer at the edge (Ed_1). The third map indicates the deployment of *blocks* on a private cloud instance at the fog (fog_1). The fourth map indicates the deployment of *blocks* on public cloud provider (Cl_1), and the fifth map indicates the deployment of *blocks* on an end-user device (EU_1).

$$\begin{aligned} Wf_1 = {} & map_1(Ed_1, blk_1) \Longleftrightarrow map_2(Ed_1, blk_2) \longleftrightarrow \\ & map_3(fog_1, blk_3) \longleftrightarrow map_4(Cl_1, blk_4) \longleftrightarrow \\ & map_5(EU_1, blk_5), \end{aligned} \quad (7)$$

The $blk_{1,\ldots,5}$ are *blocks* created by using the following configurations:

$$\begin{aligned} blk_1 &= (mem, ret_1, hub, prep_1, mem) \\ blk_2 &= (mem, ret_2, indexing, prep_2, net) \\ blk_3 &= (net, ret_3, analytics, prep_3, net) \\ blk_4 &= (net, ret_4, preservation, prep_4, net) \\ blk_5 &= (net, ret_5, visualization, prep_5, net), \end{aligned} \quad (8)$$

where, *hub* is a nanoservice for the acquisition of data from different sensors, *indexing* is a nanoservice that registers the acquired data in a database, *analytics* microservice processes the acquired data at the fog, *preservation* microservice is deployed at the cloud, and it is in charge of storing the data acquired in a cloud storage location, *visualization* is a microservice consumed by the end-users to visualize the results of the processing of data, and $prep_{1,\ldots,5}$ are the continuity schemes attached to each *block* (see Section III-B). Thus, these micro/nanoservices are created from the following maps:

$$\begin{aligned} hub &= template(mem, sc_{hub}, mem) \\ indexing &= template(mem, sc_{index}, net) \\ analytics &= VC(net, sc_{analy}, net, API) \\ preservation &= VC(net, sc_{pres}, net, API) \\ visualization &= VC(net, sc_{vis}, net, API), \end{aligned} \quad (9)$$

where sc_x represents the source code, application or function deployed by the micro/nanoservice x. Therefore, the health processing structure HWf_1 includes three elements: *i)* the Wf_1 presented in (7), *ii)* the data source (src_1), and *iii)* the data sink (snk_1), as follows:

$$HWf_1 = (src_1, Wf_1, snk_1), \quad (10)$$

where src_1 represents a link to a spirometry repository and snk_1 is a link to a storage service to preserve the processed data.

An edge-fog-cloud-EndUser structure for the continuous processing of health data, similar to the one previously described, also can be chained with other edge-fog-cloud structure by following the very same principles of the chaining of *blocks* (either microservices or nanoservices). We called this chain as *macroworkflows* (MWf_x), and it is denoted as:

$$MWf_x = HWf_n \longleftrightarrow HWf_{n-1} \longleftrightarrow \ldots \longleftrightarrow HWf_1 \quad (11)$$

B. CONTINUITY SCHEMES FOR FULFILLING NON-FUNCTIONAL REQUIREMENTS

The previous subsection introduced the architectural model for the continuous processing of health data by using edge-fog-cloud structures (focused mainly on functional requirements). The next step is to provide mechanisms for *blocks* to transparently exchange data in a secure, efficient, and reliable manners. This subsection describes a multi-continuity model for the application of non-functional requirements such as efficiency, reliability, and security, which are that critical for the management of sensitive information (i.e., health data) processed in multiple environments (any of edge, fog, cloud or end-user devices).

We recall that this model considers the building of three types of continuous schemes that are built in the form of patterns and managed as *blocks*. These schemes are:

- *Continuous security* or *CS* schemes that were created by using *blocks* linked to services for establishing access controls, data integrity verification, and data confidentiality (and privacy of their owners) when structures $edge \vee fog \vee cloud \vee EndUser$ exchange sensitive data.
- *Continuous reliability* or *CR* schemes were created by developing *blocks* that include fault-tolerance algorithms for recovering data in scenarios of failure (outages) when structures $edge \vee fog \vee cloud \vee EndUser$ exchange sensitive data.

- *Continuous efficiency* or *CE* schemes convert applications and services deployed by *blocks* in structures *edge* $\vee$ *fog* $\vee$ *cloud* $\vee$ *EndUser* into parallel patterns, which improve not only the performance of these structures but also the cost-efficiency of data storage and transportation.

In this model, the previous schemes are built in the very same form in which the *blocks* are built. This means a service developed to fulfilling a given NFR is encapsulated into a *block*, which can be coupled to other *blocks* to create comprehensive solutions for fulfilling multiple NFRs. We called these solutions as continuity schemes.

The idea is to attach a continuity scheme to each *block* in an edge-fog-cloud processing structure to enforce the fulfillment of NFRs for data arriving and departing to/from a *block* in the processing structures. These schemes are defined in coordination with the push and pull operations invoked by the *blocks* of a structure [85].

A continuity scheme is built in two versions. The first one is called outcoming schemes (*outCS*) and it has been created to add properties such as security, reliability, integrity, and cost-efficiency management to the data before sending them from a *block* to another one through any of the fog, the cloud, or end-users' devices. In our model, the algorithms that fulfill NFRs are organized as a *processing pipeline*. This pipeline is triggered each time a *Blk* invokes a push operation for transporting data to a remote location, either to the input interface of a *block* or a sink. The second version is called incoming scheme (*inCS*) and it was created to manage data prepared by an *OutCS* scheme but the *blocks* of this scheme version are invoked in an inverse manner. This means *outCS* represents the coding of departing data from a *block* to fulfilling NFRs configured for that *block*. In turn, *inCS* represents the decoding of data when valid *blocks* (or end-users) are consuming arriving data. This continuity scheme is used by *blocks* in pull operations to retrieve data.

The *inCS* schemes are executed each time a data arriving to the input interfaces of the nano/microservices (see *inCS* in Fig. 3). The *OutCS* schemes are executed each time a data is departing from a nano/microservices (see *OutCS* in Fig. 3).

In this paper, the pipelines created by using continuity schemes include microblocks (*MBlks*) such as data compressing (*comp*), data hashing&indexing (*h&i*), data coding (*cod*), data encryption (*enc*), and data uploading (*upl*). This means that the pipelines created by using *inCS* schemes include *MBlks* such as data downloading (*down*), data decryption (*decr*), data decoding (*deco*), and uncompressing (*unco*).

1) A TREE OF RECURSIVE MAPS FOR BUILDING OF EDGE-FOG-CLOUD PROCESSING STRUCTURES

The building of edge-fog-cloud structures is performed by using a tree of coupling maps (see an example of a structure built by using coupling maps in Fig. 4). This tree represents the configuration of an edge-fog-cloud processing structure. This configuration is the basis used in this model to deploy a processing structure on edge-fog-cloud environments. It also is used to ensure that functional services/systems (*blocks* associated to the data lifecycle such as analytic, preprocessing, processing, and visualization), as well as the services for fulfilling NFRs (the *blocks* included in the *outCS* and *inCS* continuity schemes such as security, reliability, and efficiency), are both invoked and executed in a well-defined sequence. This is achieved by following the coupling maps both of continuous deliver schemes of the solution (functional applications and services) as well as the continuity schemes *outCS* and *inCS* (services for NFRs).

Fig. 4 shows a conceptual representation of this tree of maps and its corresponding notation. In the first level of this tree is placed the coupling map of the functional services (*Blks*) that in this model are implemented in the form of patterns, which can be deployed on any of edge, fog, cloud or end-user depending on the execution maps of each *blk*. In this level each *block* branch produces two branches: the first one to invoke non-functional continuity schemes (see *InCS* and/or *outCS*). The second one is the microservice/nanoservice that will execute the functional application associated with the "father" *Blk* defined in its execution map. The *inCS/outCS* branch produces a new level based on the coupling map for the NFRs *blocks* (see $MBlk_{pattern}$), which is a recursive representation of the first level of the tree. This means that each NFR *blk* in a $MBlk_{pattern}$ produces again two branches, the first one for NRF continuity schemes and the second one for the microservice/nanoservice. This creation of new levels continues until finding a *blk* that does not invoke a *inCS/outCS*, which meaning the leaves of the tree (*microservice/nanoservice*) have been found.

C. DEVELOPING CONTINUITY SCHEMES FOR FULFILLING NFRs IN REAL HEALTH DATA PROCESSING SCENARIOS

This section presents the development of continuity schemes (see *inCS/outCS* of the continuity schemes in Fig. 4) for edge-fog-cloud processing scenarios. These schemes were built by using a tree of recursive maps described in the previous section.

Three continuity schemes suitable in real health scenarios were developed by using security (*CS*), reliability (*CR*), and cost-efficiency storage applications (*CE*). These schemes were implemented in the form of parallel patterns (see $MBlk_{pattern}$ in Fig. 4). These schemes were incorporated into the prototype used for the experimental evaluation.

1) *CE*: EFFICIENCY CONTINUITY SCHEME BASED ON PATTERNS

The continuity schemes were implemented as parallel patterns by following the very same notation of $Blks_{patt}$ described in Section III-A2 to build a parallel pattern called $MBlk_{pattern}$.

The *CE* scheme was defined to speed up the performance of any *block* (*blk*) in a solution. A *MBlk* could be part of a more complex pattern to improve the performance of the data exchange performance. This can be performed by using

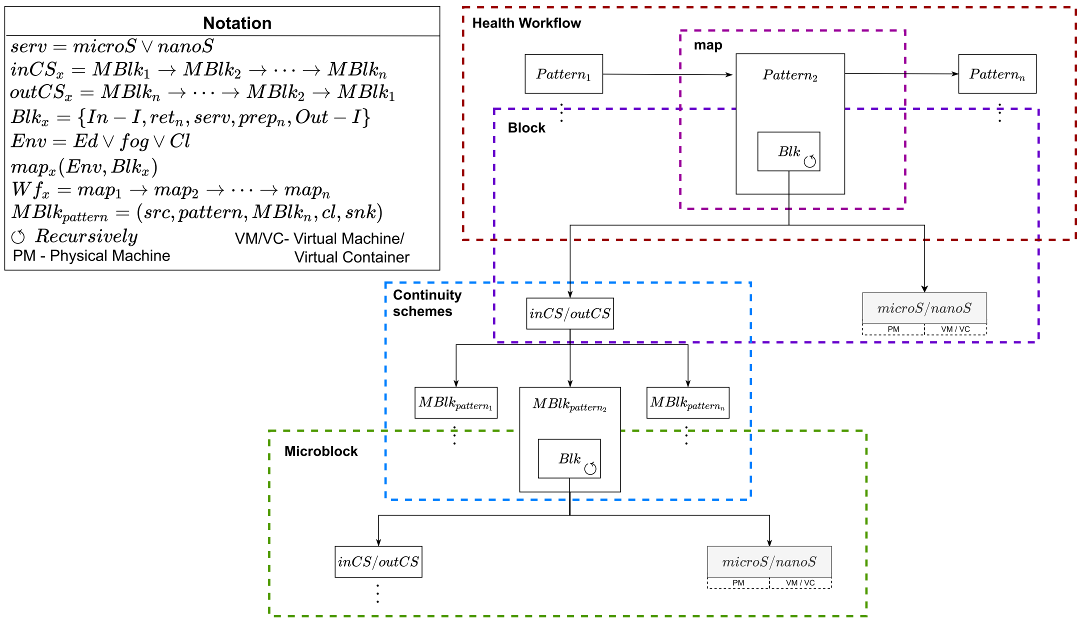

Notation
$serv = microS \vee nanoS$
$inCS_x = MBlk_1 \rightarrow MBlk_2 \rightarrow \cdots \rightarrow MBlk_n$
$outCS_x = MBlk_n \rightarrow \cdots \rightarrow MBlk_2 \rightarrow MBlk_1$
$Blk_x = \{In - I, ret_n, serv, prep_n, Out - I\}$
$Env = Ed \vee fog \vee Cl$
$map_x(Env, Blk_x)$
$Wf_x = map_1 \rightarrow map_2 \rightarrow \cdots \rightarrow map_n$
$MBlk_{pattern} = (src, pattern, MBlk_n, cl, snk)$
Recursively
PM - Physical Machine
VM/VC- Virtual Machine/ Virtual Container
Health Workflow
map
Block
Continuity schemes
Microblock
$Pattern_1$
$Pattern_2$
$Pattern_n$
Blk
$inCS/outCS$
$microS/nanoS$
PM
VM / VC
$MBlk_{pattern_1}$
$MBlk_{pattern_2}$
$MBlk_{pattern_n}$

FIGURE 4. Tree of *blocks* and coupling maps for deploying edge-fog-cloud processing structures.

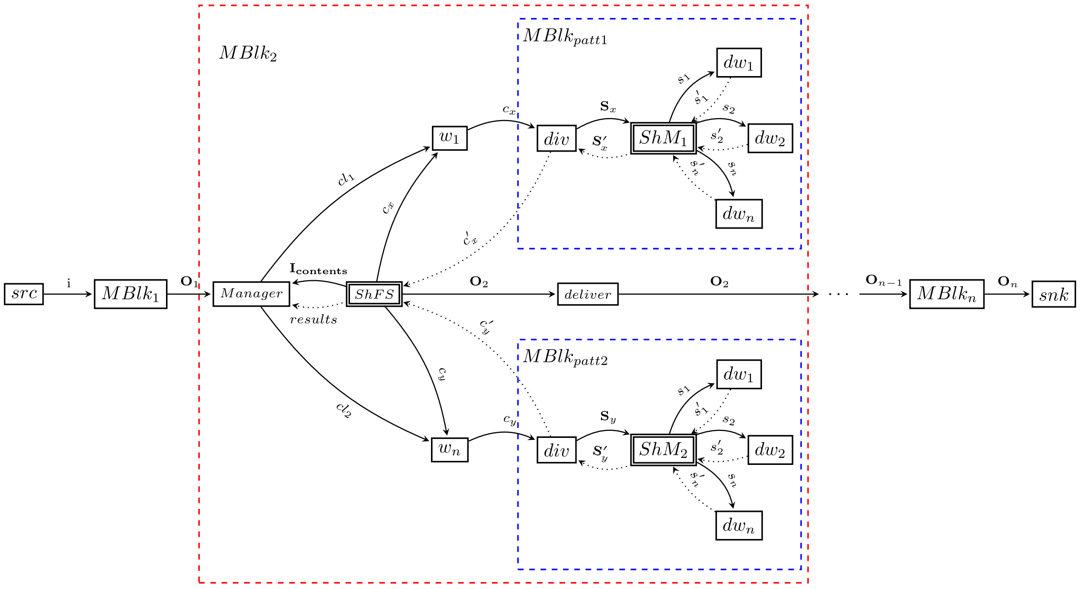


FIGURE 5. Maps of the reliability *MBlk* implementing a combination of patterns.

the recursive feature of this model (see microblock level in $MBlk_{pattern}$ branch depicted in Fig. 4).

Fig. 5 shows an example of a pipeline pattern of a *CR* scheme. This scheme was created by using the tree of recursive maps described in the previous section. This pipeline includes a set of *MBlks* ($MBlk_1, MBlk_2, \ldots, MBlk_n \in$ edge-fog-cloud structure). The $MBlk_2$ considers a pipeline pattern built by three *microblocks* such as the manager, *ShFS* (a shared resource pattern that can be linked to any of file system, partition, or cloud location) and a data *deliver*. This scheme reuses the *ShFS* pattern as in-memory storage (ShM_1 and ShM_2) for improving the exchange of data of the *blocks* using this scheme.

In this example, the $MBlk_2$ pattern acquires incoming data ($\mathbf{O}_1$) from $MBlk_1$, transforms data ($\{c_1, c_2, \ldots, c_n\} \in \mathbf{O}_1$) into information and delivers information ($\mathbf{O}_2$) to the next $MBlk_n$. The manager in this pattern invokes another pattern (a traditional *manager/worker*), where the manager sends tasks ($\{cl_1, cl_2, \ldots, cl_n\} \in \mathbf{O}_1$) to a set of workers ($\{w_1, \ldots, w_n\}$). The workers receive tasks from the manager and gets the data associated to the tasks ($\{cl_1, cl_2, \ldots, cl_n\} \in \mathbf{O}_1$) from *ShFS*. Each worker invokes a third pattern: a *divide&conquer* (*D&C*), which results in the execution of $MBlk_{patt1}$ and $MBlk_{patt2}$ patterns. Each *block* (*Div*) splits the contents sent by the workers into *s* segments ($\{s_1, \ldots, s_n\}$). The segments then are delivered to the workers of the *D&C* ($\{dw_1, \ldots, dw_n\}$) through local in-memory shared resources (ShM_1 and ShM_2). The segments processed by each *dw* are sent to the *divs* through the shared in-memory resource patterns (ShM_1 and ShM_2). The *deliver* can retrieve the resultant data ($\mathbf{O}_2$) from the shared pattern (*ShFS*) and then delivers it to the next *MBlk* in the original pipeline pattern.

2) *CR*: CONTINUOUS RELIABILITY SCHEME FOR EDGE-FOG-CLOUD PROCESSING STRUCTURES

A continuous reliability scheme ($MBlk_{idaDQ}$) was created in the form of *divide&conquer* pattern, to provide data (medical images, spirometry studies, and backups of digital ECGs) with fault-tolerance properties. This pattern implements a coding *Mblk* ($MBlk_{cod}$) that executes an application based on the IDA algorithm [134], [135]. This app splits the data into *n* segments including redundancy for *m* segments are sufficient to recover the original whenever $n > m$, which allows $n - m$ cloud/fog outage scenarios. The codification scheme ($OutCS \in MBlk_{cod}$) splits a file into the segments, adds redundancy to data for fault tolerance, and transports the encoded data to other *MBlk*. The decoding of data ($InCS \in MBlk_{cod}$) scheme includes a $MBlk_{deco}$ to recover the original data.

The parallel patterns described in Section III-A2 were used to implement $MBlk_{cod}$, and the $MBlk_{deco}$ in an efficient manner. The $MBlk_{cod}$ is implemented in the form of a *manager/worker* pattern, where each worker is a clone of the IDA algorithm which is implemented as a memory-based *divide&conquer* pattern. In this way, the notation to build $MBlk_{cod}$ is as follows:

$$MBlk_{idaDQ} = (input, dq, MBlk_{ida}, 5, output) \quad (12)$$
$$MBlk_{cod} = (input, mw, MBlk_{idaDQ}, 3, network) \quad (13)$$

$MBlk_{idaDQ}$ is denoted in (12), which reads data from the *input* directory and implements the IDA algorithm as a *divide&conquer* pattern including five workers. In turn, (13) denotes the construction of the *manager/worker* pattern by using three workers, which invoking three replicas of the $MBlk_{idaDQ}$. $MBlk_{deco}$ is implemented as a *manager/worker* pattern. The *Manager* reads the directory of the downloaded segments, creates a list of all contents, and sends the tasks to the workers, which retrieves the contents to be processed. Each *worker* decodes the content previously encoded by $MBlk_{cod}$.

3) *CS*: CONTINUOUS SECURITY SCHEME FOR ENSURING OF IoT DATA

The *CS* schemes mitigate security risks when an edge-fog-cloud processing structure manages sensitive data. These schemes include a *Blk* built by using a pipeline of three security services encapsulated into *MBlks*. The first $MBlk_1$ encrypts data by using a symmetric cryptosystem based on AES [136]. The confidentiality of access controls is managed by $MBlk_2$, whereas the last one ($MBlk_3$) ensures the integrity of data by using a digital signature and a secure envelope created by using CP-ABE algorithm [137], [138].

The *outCS* and *inCS* version of the *CS* scheme were created as encryption and decryption pipelines respectively. The three *blocks* of these pipelines ($MBlk_1$, $MBlk_2$ and $MBlk_3$) were developed in the form of a *manager/worker* pattern (described in section III-A2) to speed up the performance of their encryption/decryption *blocks*. *outCS* is executed before sending data outside of a *block*, whereas the *inCS* is executed immediately before data are retrieved from the remote location.

D. ARCHITECTURE AND DESIGN PRINCIPLES

This section describes an architecture that materializes the model for building edge-fog-cloud processing structures. This architecture was developed as a service composition based on patterns of micro/nanoservices to achieve two main goals: *i)* to enable the deployment of the edge-fog-cloud structures on the subjacent infrastructure; and *ii)* to enforce the controls defined for each *block* of each processing structure.

It is important to note that this architecture has been designed to manage all edge-fog-cloud processing structures as well as each *block* considered in the structure as a *service*. This architecture builds these services by using the trees of maps created for each processing structure.

Four types of *requests* are supported in this architecture: *i)* creating services from *blocks* available in the service mesh; *ii)* launching a processing structure as a service; *iii)* delivering data to services; and *iv)* recovering raw data and results produced by services.

Fig. 6 depicts the stack architecture for the management of data proposed to materialize the model of building of edge-fog-cloud processing structures. As can be seen, the stack considers the following five layers:

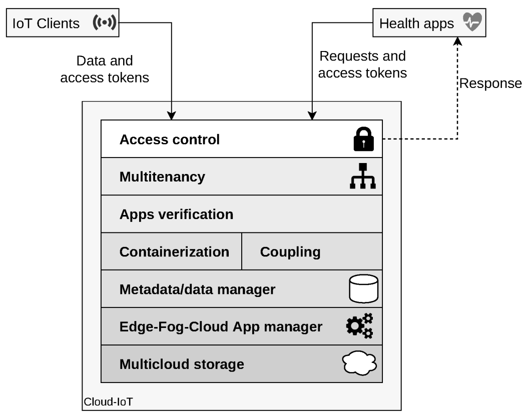


FIGURE 6. Stack architecture of the IoT-edge-fog-cloud management service.

- **Access control**: this layer represents a front-end to manage incoming requests performed by external agents (IoT clients and health apps). The access is performed by using an authentication system that validates the tokens provided by external agents. This simple tokenization system can be changed by a system based on private/public keys.
- **Multi-tenancy**: is a layer that establishes isolation for users validated in the previous layer. Service projects are associated with users for creating privacy as it is assumed that some services will be used by multiple users. This avoids unauthorized access to both, the services as well as the data and results produced by these services.
- **Edge-fog-cloud app manager**: this layer manages the applications executed by each service ($serv = MicroS \vee NanoS$) associated with each service by using the tree of maps (see Section III-B1).
- **Metadata/data management**: this layer manages the maps defined in this model such as execution, coupling, patterns, and environment maps. This layer includes a mechanism for verifying of the continuous delivery of data and continuity schemes.
- **Multicloud storage**: this layer contains a content delivery network for supporting the delivery and retrieval of data/results to/from each service managed by this architecture using a layer of multi-cloud storage.

The architecture includes client software for end-users to interact with processing structures. The IoT clients collect data produced by different IoT sensors and send the collected data to edge-fog-cloud service. The health apps are in charge of receiving from the architecture the results processed by the services. This client also can react depending on the received information (i.e., to generate notifications to physicians or nurses).

Fig. 7 depicts a stack architecture of both types of applications sending requests to edge-fog-cloud services. The IoT clients are shown on the left side of this figure, whereas the health apps are depicted on the right side. The IoT client stack is deployed on the edge, and it considers five layers: the front-end, access, data hub, credentials management, and outcoming continuity layers. The front-end layer receives the data produced by the sensors through a network interface. The access layer manages the credentials for the IoT devices presented/created. The data hub layer creates tokens that are associated with the packets sent by each IoT device. The management of the received data is associated with a valid IoT device in the credentials management layer. Finally, the outcoming continuity layer implements an *outCS* scheme to prepare the data before to be sent to either the fog or the cloud (represented by IoT cloud).

Health apps are connected to an architectural stack deployed on either the cloud or the end-user side. This stack also includes an access layer that manages the incoming requests sent by different users. The credentials management layer is in charge of managing the users and applications tokens. The app layer associates the tokens of the users/apps to a corresponding and valid edge-fog-cloud service. The metadata/data management layer process the execution map of the microservice (health app) for getting access to the source data, execute the application, and deliver results in a data sink. The outcoming/incoming continuity layer implements a *outCS* and *inCS* continuity schemes assigned to the health app (generally defined by the user of the app).

At this point, it has been shown how the applications used by the clients of the edge-fog-cloud service have been organized to send data and to receive results to/from a service. Fig. 8, in turn, shows the architecture stacks for the services ($serv = MicroS \vee NanoS$) of all the environments where a workflow could be deployed. In this way, engineers can develop services based on edge-fog-cloud processing structures by following the protocol established by the architecture stacks. For instance, services can be designed to acquire data through the cloud from the IoT devices to deliver them to

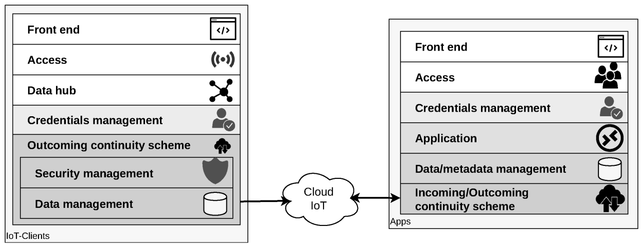


FIGURE 7. Stack architecture of the IoT clients and health applications to interact with the edge-fog-cloud services (cloud-IoT).

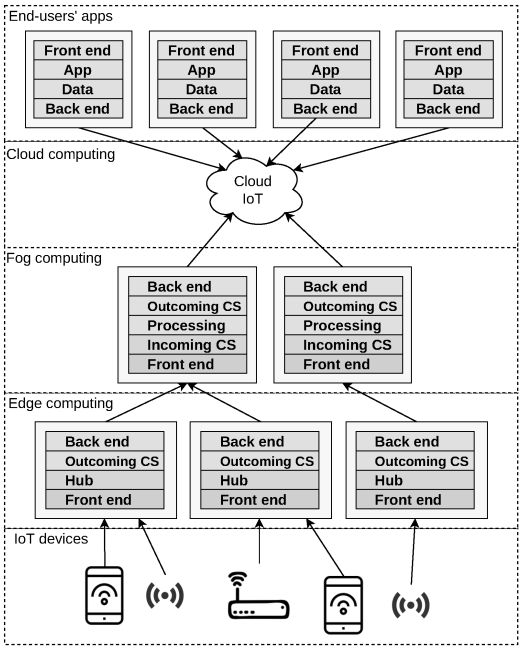


FIGURE 8. Architecture stack of edge-fog-cloud structures management service.

end-users' devices by following the front and back ends of the stacks available in the architecture for each environment.

IV. EXPERIMENTAL EVALUATION: A CASE STUDY BASED ON DATA PROCESSING IN AN EMERGENCY ROOM

An experimental evaluation for measuring the performance of the architectural model proposed in this paper has been conducted in the form of a case study. This case study considers four experimental scenarios.

The *first scenario* considers experiments where health care heterogeneous data (i.e., vital signs, and activity level) are acquired from different sensors and applications to be transported from the edge to the cloud through multiple hops. This scenario is suitable as industry and research community have identified the Internet of Things (IoT) as a key enabling technology to radically improve health care services and applications such as ambient assisted living (AAL) and remote patient monitoring [139].

In the *second scenario*, the attention was focused on edge-fog-cloud service created for the continuous monitoring of electrocardiogram (ECG) data of users. These data were collected by IoT devices and were used for detecting QRS complexes. This service also considers the creation of alerts for both end-users and physicians. This evaluation scenario is relevant as continuous monitoring of health parameters allows patient care teams to improve the early tracking of upcoming adverse episode indicators (i.e., cardiac arrest) for timely treatment and intervention when needed. Particularly, centralized monitoring systems help to enable predictive analytic approaches, which are expected to play a key role to early recognize patient deterioration [140]. One of the most relevant health parameters is the ECG, which is widely adopted as a diagnostic tool that enables the diagnosis of minor to major health risks [141].

The *third scenario* is focused on evaluating multiple services producing dataflows of health IoT data through edge-fog-cloud environments consumed by multiple health care professionals. This scenario is because medical staff could require different tests and studies of patients coming from heterogeneous sources to improve the diagnosis process in an emergency room.

The *fourth scenario* considers the creation of continuous delivering, through the cloud, of health data to external health care professionals for health care professionals getting second opinions. In this scenario, simultaneous edge-fog-cloud services were created for continuous monitoring of ECG data, transporting spirometry studies [142], and transporting tomography studies [143], which could be consulted by health care professionals in the emergency room and external professionals by using health apps.

In addition, an *evaluation study* was conducted to the assessment of the impact of edge-fog-cloud processing structures on the storage utilization and the computation complexity of big health data solutions built by the model proposed in this paper.

A. TEST ENVIRONMENT

Table 2 shows the main characteristics of the servers, virtual machines, and containers used to conduct the experimental evaluation. EC2-1 is a virtual machine deployed on Amazon, whereas Compute1 and Compute2 are physical servers used as fog nodes (Mexico). And, the workstation is a personal computer at the edge used as a sink where the sensors push the collected data.

TABLE 2. Characteristics of the infrastructure used for experimentation.

Compute	Description	Cores	RAM (GB)	Storage	OS
EC2-1	Cloud node	16	32	600 GB	Ubuntu 18.04
Compute1	Fog node	12	64	2.7 TB	Centos6
Compute2	Fog node	6	12	256 TB	Centos6
Compute3	Fog node	12	64	256 TB	Centos6
Workstation	Edge node	4	16	2 TB	Ubuntu 18.04

B. SCENARIO I: CONTINUOUS ACQUISITION/ MONITORING OF HEALTH CARE IoT DATA

Continuous monitoring of health parameters allows patient care teams to improve the early tracking of upcoming adverse episode indicators (i.e., cardiac arrest) for timely treatment and intervention when it is needed. Particularly, centralized monitoring systems help to enable predictive analytic approaches, which are expected to play a key role to early recognize patient deterioration [140]. One of the most relevant health parameters is the ECG, which is widely adopted as a diagnostic tool that enables the diagnosis of minor to major health risks [141].

1) CONFIGURATIONS, EXPERIMENTS, AND METRICS

In the experimental deployment, IoT devices were connected to an edge node (*Hub*) through an IEEE 802.15.4 WSN network. Each IoT device was implemented in a CC2650 Sensortag with the IEEE 802.15.4 stack of the Contiki OS. The WSN was based on the IEEE 802.15.4 non-beacon enabled mode with unslotted CSMA/CA protocol, which is operated in the unlicensed 2.4 GHz ISM band. Thus, the inter-arrival time of data packets in the edge node will fluctuate with the number of active IoT devices. In addition, according to the CSMA/CA MAC protocol, an IoT device could discard packets when the wireless medium remains occupied with active transmissions for a period determined by the exponential back-off algorithm [144]. Moreover, two or more transmission could collide when nodes select the same back-off timer.

The edge node is a physical server at CICESE Monterrey (a research center at northeast state of Mexico) with two network interfaces, Ethernet and IEEE 802.15.4. Particularly, the Ethernet interface is used for the IoT cloud connection, and the IEEE 802.15.4 transceiver was used for WSN communication (see Table 2). The IoT cloud is a physical server in a datacenter at another state in Mexico (Tamaulipas state at the northeast). ECG nodes are IoT devices deployed on a star topology, where the sink is the edge node connected to the cloud through an Ethernet interface. In this evaluation, each wireless node continuously transmits a 1-lead ECG signal.[3] This ECG was obtained from an online synthetic data library. The obtained ECG data were sampled at 500 Hz with 12 bits per sample.

Three different configurations were performed to evaluate the *block* of data acquisition:

1) Low packet rate (LPR): IoT devices transmitting 14 packets per second (pps), with 55 ECG samples each (i.e., 110 Bytes of payload per packet).
2) High packet rate (HPR): IoT devices transmitting 128 pps, with 16 ECG samples each (i.e., 32 Bytes of payload per packet).
3) Heterogeneous: half IoT devices with 128 pps and half IoT devices with 14 pps.

a: EXPERIMENTS

The experiments performed with these configurations consider that each minute was deployed one IoT device transmitting ECG data packets until it reaches eight IoT devices. That is, in the first minute, there is one IoT device working, and at the final slot (the beginning of minute eight) eight IoT devices are transmitting ECG data packets. Each IoT device is turned on and starts transmitting 14 pps for configuration 1 and 128 pps for configuration 2, respectively. When an IoT device is turned on, it remains on until the end of the experiment. For the heterogeneous scenario (configuration 3), the first IoT device starts transmitting a packet rate of 128 pps, and the following IoT devices alternate the packet rate between 14 and 128 pps. The experiment duration of three scenarios was eight minutes.

b: METRICS

Two different sets of metrics were used in this performance evaluation. One set is used at the edge node to measure the performance of the WSN network. And, the second set of metrics is used at the IoT cloud to measure the response time (performance) of the architectural model. Note that the experiments measure the impact of load increment on the response time of the services. Therefore, all metrics were calculated considering the ECG data of the first IoT device turned on. The rest of the IoT devices were used to generate load for the services.

The *server response time (SRT)* is defined as the average response time of the server (any of the fog or the cloud) to attend a portion of packages received at the edge node. The SRT is calculated as:

$$SRT = \frac{1}{N_{pt}} \sum_{i=2}^{N_{pt}-1} pt_i - pt_{i-1} \tag{14}$$

where pt_i is the arrival time of the $i-th$ portion of packages at the server, pt is a portion of packages received in the same request from the edge node to the server, and N_{pt} is the number of portions received.

The *packet delivery ratio (PDR)* was measured to evaluate the performance of the network where the *block* of the WSN was placed at. The PDR is the average number of ECG data packets that are successfully received at the edge node. It is calculated as:

$$PDR = \frac{ECG_{rx}}{ECG_{tx}} \tag{15}$$

where ECG_{rx} is the number of data packets received at the edge node and ECG_{tx} is the number of ECG data transmitted for IoT devices.

The *inter-arrival time (IAT)* is the average arrival time between packets successfully received at the edge node. The average IAT provides a measure of the arrival time variability by each packet and it is calculated as:

$$IAT = \frac{1}{N_{rx}-1} \sum_{i=2}^{N_{rx}-1} x_i - x_{i-1} \tag{16}$$

where x_i is the arrival time at the edge node of the i_{th} packet, x_{i-1} is the arrival time at the edge node of the previous packet, and N_{rx} is the total number of ECG data packets received from the first IoT device at the edge node.

2) EVALUATION RESULTS

This subsection presents the evaluation results of the phase I. Fig. 9 and Fig. 10 present the set of WSN metrics of PDR and IAT when increasing the number of IoT devices. Recall that IoT devices contend for the channel access in the deployed WSN (see Section IV-B1). Thus, the greater number of contenders, the less probability of accessing the channel, which reducing the network performance.

As it can be seen in Fig. 9, the PDR of low packet rate (LPR) configuration (55 samples) performs better than the other two configurations because each IoT device generates a significantly lower number of packets per second, 14 pps in LPR configuration versus 128 pps in the high packet rate configuration. This means the contention and packet losses are lower in LPR configuration.

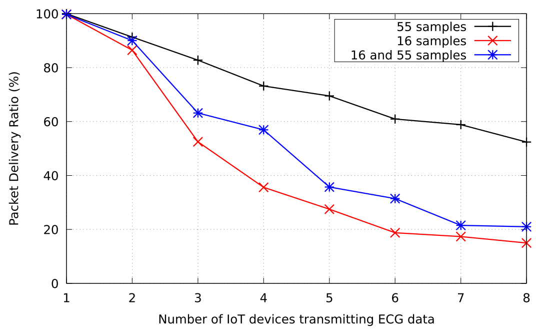


FIGURE 9. **Packet delivery ratio (PDR) obtained at the edge node as the number of transmitting IoT devices increases for low packet rate (55 samples), high packet rate (16 samples), and heterogeneous (16 and 55 samples) configurations.**

[3]The design and evaluation of ECG data collection are beyond the scope of this paper.

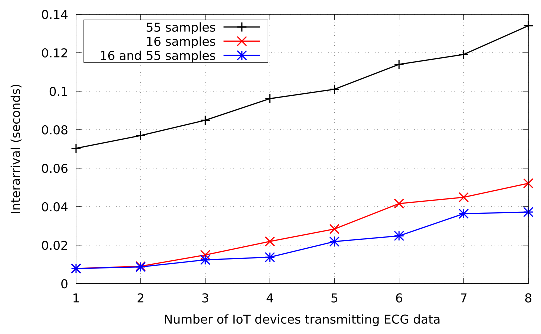


FIGURE 10. **Inter-arrival time (IAT) obtained at the edge node as the number of transmitting IoT devices increases for low packet rate (55 samples), high packet rate (16 samples), and heterogeneous (16 and 55 samples) configurations.**

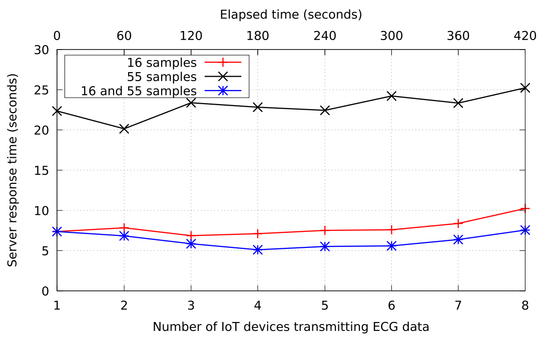


FIGURE 11. **Response time observed by the edge node when a *block* of packages is sent to the cloud node as the number of transmitting IoT devices increases for low packet rate (55 samples), high packet rate (16 samples), and heterogeneous (16 and 55 samples) configurations.**

Conversely, the IAT is higher for LPR configuration, because packets in LPR configurations are significantly larger, 110 bytes than in high packet rate (HPR) configuration, 32 bytes. Thus, as the air time in LPR configurations is larger, the IAT is also higher. It is worth noting that metrics, PDR, and IAT, for the heterogeneous configuration are very similar to HPR configuration. It can be explained because IoT devices transmitting at a higher packet rate capture the channel access.

Fig. 11 presents the server response time (SRT) observed by the edge node when a portion of packages is sent to the cloud. Every two seconds a bot, programmed at the edge node reads the ECG traces, generates groups of ECG packets that are sent to the fog, where a proxy redirects the groups to a processing worker. Also, the groups size is closely related to the packet IAT at the edge node. Thus, as can be seen in Fig. 11, the response time increases with the group size.

Results in this subsection show that the architectural model can be used to enable a continuous monitoring system for health parameters as ECG. The results highlight that an end-to-end approach design is required for health care applications, showing also that a design consideration in the first stage (i.e., packet generation rate), could heavily impact the system performance.

C. SCENARIO II: EDGE-FOG-CLOUD SERVICE FOR CONTINUOUS ECG MONITORING

In this section, we describe a prototype based on an edge-fog-cloud service created for the continuous monitoring of wireless ECG (IoT devices). The *blocks* of this service detect QRS complexes and create alerts for both end-users and physicians. This section also presents the methodology to evaluate this service and the results of such an evaluation.

1) CONFIGURATIONS AND EXPERIMENTS

This edge-fog-cloud service was implemented in a prototype to create continuity schemes (*CS*-security, *CR*-reliability, and *CE*-efficiency). These schemes prepare ECG data each time these data are transported from the edge to the fog or the cloud. Different wireless ECG monitoring systems deployed in an emergency room were considered for different patients could be simultaneously monitored. The devices were connected to an edge node through a wireless sensor network (WSN).

The edge-fog-cloud service showed in Fig. 12 was used for the management and processing of these health data. In this service, the IoT device sends signals to a nanoservice at the edge, which includes an application called *hub*. This *block* writes the acquired data into a file for each IoT device that is placed in a given directory. This directory is processed by using a *manager/worker* pattern, where a *block* called *monitor* (the manager) acquires the files (**F**). These files are associated with a token and credentials that are distributed to *n workers* by using the two-choices algorithm [145], [146] for load balancing purposes.

This type of distribution creates n subsets of files (f_i) as well as one thread per worker. Each *worker* thus reads a f_i file and transforms it into a JSON format (**D**). The JSON files are sent to a *block* in the fog called *proxy*, which is in charge of distributing the load between n *blocks* ($processing_1, processing_2, \ldots, processing_n$). Each *block* processes the data with the *QRS* complex detector algorithm [147]. *QRS* complexes and processed data are stored in a database ($\mathbf{D}^+$).

A *block* in this edge-fog-cloud solution executes a publication-subscription model (*Pub*/*Sub*) that makes the data and *QRS* complexes available for end-users to consume them. This *block* has been implemented by using an IoT Pub/Sub traditionally used to send data either to the fog or the cloud [148], [149]. This pub/sub *block* sends new notifications to subscribed users, which are generated by a *block* analyzing *QRS* and registering events in a database. Finally, a visualization *block* was developed to make available, both the alerts and the data processed for health apps, to end-users through the cloud. Fig. 13 shows an example of the visualization of ECGs.

2) EXPERIMENTAL RESULTS

Fig. 14 shows, in vertical axis, the response time of the continuous processing of ECG data, as well as the creation

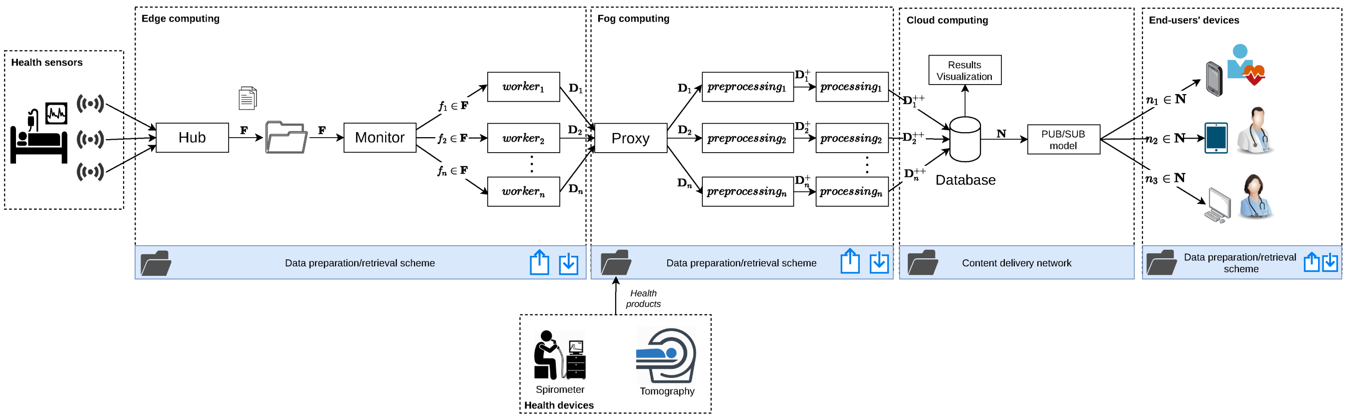


FIGURE 12. **Conceptual design of the edge-fog-cloud workflow used in this case study.**

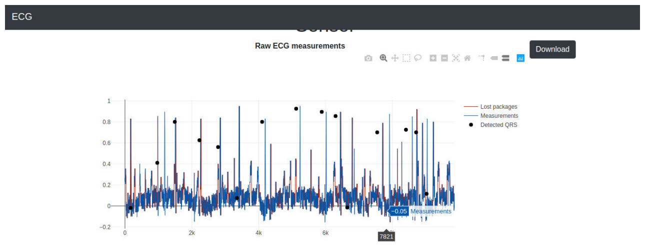


FIGURE 13. **ECGs visualization *block*.**

of alerts for different number of samples (10 in Fig. 14a, 100 in Fig. 14b, and 1000 in Fig. 14c). These response times were captured when the *blocks* use a different number of clones/workers in the patterns launched the service (horizontal axis). As expected, the time spent in the processing of data is increased with the number of samples. Nevertheless, it was also observed that the information available for decision-making processes increases with the amount of processed data. In this context, the usage of implicit parallel patterns of the services improves the performance observed by end-users. In terms of time, the patterns enable any of the medical staff, end-users, or health applications, to get more information in less time in comparison with a single solution. For instance, Fig. 14c shows that medical staff could receive, in a secure and reliable manner, the EGCs graphs and alerts of 1000 samples when the workflow is using 14 workers, whereas in the same time a traditional solution (one worker) only processes 100 samples in the same period.

D. SCENARIO III: SIMULTANEOUS HEALTH DATA PROCESSING SERVICES FOR MULTIPLE HEALTH CARE PROFESSIONALS

To conduct the third scenario of this case study, three edge-fog-cloud services were developed for transporting contents from continuous monitoring of ECG data, spirometry studies, and tomography images to the medical staff and end-users' devices as shown in Fig. 15.

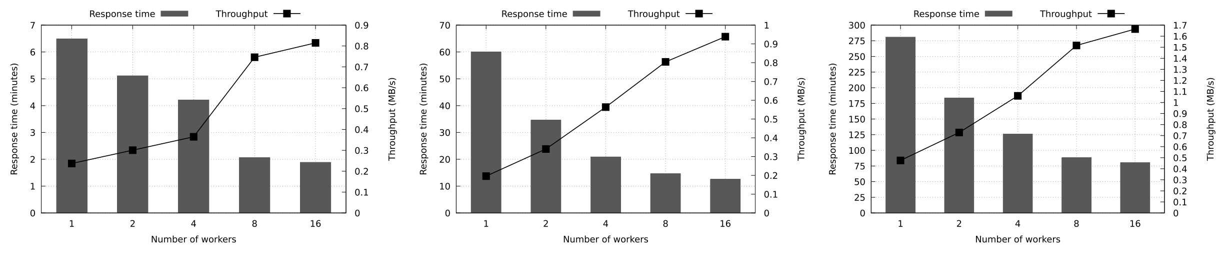
The *first edge-fog-cloud service* manages the continuous monitoring of ECGs and QRS, as well as the alert processing, is shown on the top of Fig. 15. These *blocks* and the dataset processed by this service were described in Section IV.

The *second service* starts in a data source including a repository of spirometry studies. Each spirometry study is managed as a compressed file including 10 tests with spirometry data, Kinect imagery, calibration data, and evaluation of the tests [150]. This source is connected to a service, which includes outcoming/incoming continuity schemes. These schemes add to the spirometry studies properties such as security, reliability, and efficiency in a continuous manner (*CS*, *CR*, and *CE* respectively). The outcoming continuity scheme *CS* includes *blocks* of privacy and confidentiality based on cryptography-based access controls (previously described at Section III-C3). The scheme *CR* includes a *block* of fault-tolerant (see the description of the IDA reliability algorithm at Section III-C2). The scheme *CE* includes *blocks* of compression and deduplication to reduce the amount of data being transferred by this service. The data processed by the continuity schemes are sent to the cloud where are retrieved by health applications (described at Section III-D). These health applications execute the incoming version (*inCS*) of the $CS + CR + CS$ schemes. These schemes enable medical staff to retrieve spirometry studies and get access to their content.

The third service is similar to the second one but processing medical tomography images.

These scenarios could arise during the patient stay in the emergency room, where these health contents would be required for improving the diagnosis process. The service collects the contents and delivers them automatically to medical staff in an emergency room in a secure and reliable manner.

Table 3 shows the characteristics of the two datasets used to evaluate the performance of second and third services. The first one is a dataset of spirometry contents of ten patients [150]. Whereas, the second one is a set of computer tomography images of 44 patients [152].



(a) 10 samples. (b) 100 samples. (c) 1000 samples.

FIGURE 14. Response time for the processing of ECG sensors traces varying the number of workers in the pattern.

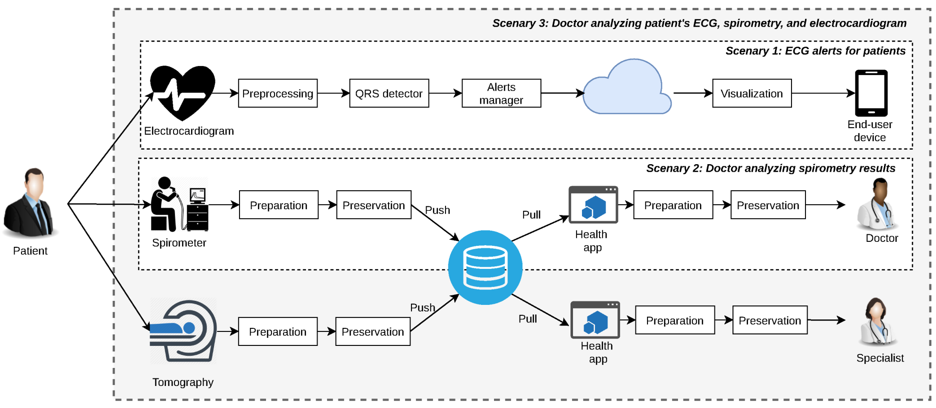


FIGURE 15. Simultaneous edge-fog-cloud services in an emergency room.

TABLE 3. Datasets used in the experimentation.

Dataset	Type	Size	No. Files	Avg. file size
spirometry	Compressed file with spirometry data, kinect imagery, calibration data, evaluation metrics, and results from an analysis of data [150].	50 GB	10	5.27 Gb
Tomographies	Imagery in DICOM format [151].	8.9 GB	23335	397.21 MB

a: EXPERIMENTS AND CONFIGURATIONS

The performance of these services was evaluated by experiments where the services were added to the edge-fog-cloud solution in a gradual manner by performing the following experiments:

1) The service processing the tomography image repository was evaluated in a separated and isolated manner.
2) The service that process spirometry studies was added to the solution and both services were evaluated when processing contents in a simultaneous manner.
3) An experiment adds the three services to the edge-fog-cloud solution, which simultaneously executes the three services.
4) An experiment was performed by comparing the edge-fog-cloud solution with a workflow engine available in the state-of-the-art called DagOnStar [153].

1) EXPERIMENTAL RESULTS

Fig. 16 shows the response times produced by a service when processing medical tomography images. In this experiment, 44 images were prepared for making them available for medical staff to retrieve them in a secure and reliable manner. As can be seen, the impact of the efficiency continuity scheme (*CE*) used in this service on the solution performance is evident. The patterns of the *CE* scheme using 12 workers could deliver the secured images in 10 minutes, which was one magnitude order better than a traditional service using a single worker. This behavior was also observed when the service processing spirometry studies.

Fig. 17 shows the response times of two simultaneous services (one processing tomography images and the other spirometry studies). As can be seen, the efficiency of the patterns is preserved by the architectural model even when processing heterogeneous contents by simultaneous services. The main difference between these results and the previous ones is that the peak of performance improvement is achieved earlier. In this scenario, the pattern produces overhead on the response times because very large contents (5.2 GBs per each spirometry study) are processed, which produces

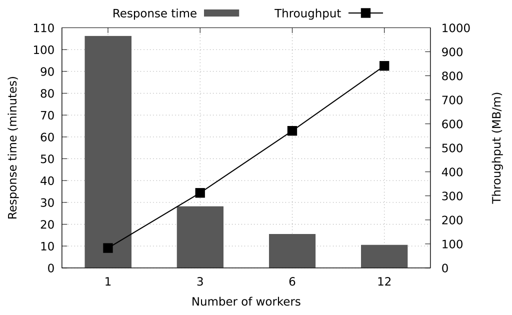


FIGURE 16. Response time and throughput observed when processing medical images.

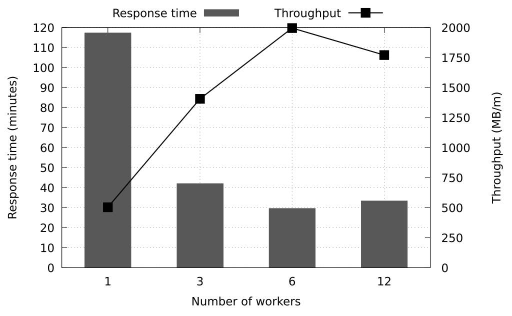


FIGURE 17. Response time and throughput observed when processing tomography images and spirometry studies.

delays when the *blocks* process "small" contents (300 MB per tomography image).

Fig. 18 shows the results of three services previously described, which retrieve all data of one patient from three different data sources (ECGs alerts warnings, spirometry studies, and tomography images). This edge-fog-cloud service processed 8.5 GB of health data in 15.46 minutes. This time includes the expedient collection plus the time spent by the outcoming continuity scheme to sent the data in secure and reliable manners to the fog, where these contents were preserved in local storage. This time also includes the time spent on a health application to retrieve these contents and to develop a *block* created for a decision-making process.

a: PERFORMANCE COMPARISON WITH A SOLUTION OF STATE-OF-THE-ART WORKFLOW

A performance comparison of the architectural model proposed in this paper with DagOnStar [153], a state-of-the-art solution, was performed as the last experiment of this experimental evaluation scenario.

DagOnStar is also based on the parallel processing of data. Note that in the qualitative comparison (see Section II), DagOnStar showed similar properties to the architectural model proposed in this paper. It is important to note that DagOnStar does not include a continuous processing approach. Therefore, only the outcoming *block* is considered in this comparison. Supposedly, there is a version of DagOnStar that implements this characteristic, but it is not published.

Fig. 19 shows that the architectural model produces a better performance than DagOnStar when processing spirometry studies (36.80%) and the tomography dataset (33.12%). This is significant, as it would expect to achieve this improvement for each *block* of each service created by the architectural model proposed in this paper.

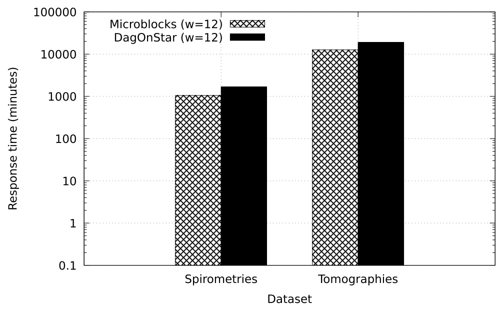


FIGURE 19. Response time for the processing of the tomography and spirometry dataset by using the proposed approach and DagOnStar.

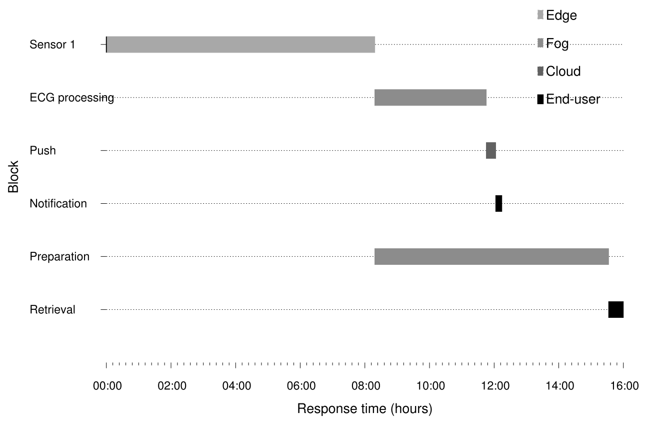


FIGURE 18. Timeline for the acquisition of ECG data and their processing, plus the outcoming continuity scheme of spirometry studies and tomography for one patient.

E. SCENARIO IV: HEALTH IoT DATA IN EDGE-FOG-CLOUD COMPUTING FOR MULTIPLE USERS

This section presents a big picture of the events that arise in the execution of edge-fog-cloud solutions tested in the previous scenario but delivering data to external health professionals by using a public cloud.

Medical inter-consultation for patients is a regular activity in general hospitals today. For instance, in the emergency room, patients are admitted and evaluated by a specialist in emergency medicine. However, in different cases, a second opinion from other specialists could be required. In these scenarios, the required specialist could be in any of the same hospital, another specialty hospital, or even in house (in extraordinary cases).

In this scenario, studies, medical images, and electronic health records of the patient must be shared with the corresponding specialist to provide her with the tools for the diagnosis. Moreover, due to health data management laws [154], [155], the studies not only must be ensured by continuity schemes to fulfill NFRs but also must be stored for a large period.

1) EXPERIMENTS AND CONFIGURATIONS

Four experiments were conducted to show the feasibility of using edge-fog-cloud solutions to support this type of scenario: supporting medical and inter-consulting for patients preserving health data. In this experimental scenario were evaluated the very same services described in the previous experimental scenario (see Section IV-D).

The following are the experiments performed:

1) Experiment I: the preparation of a tomography image of one patient, which was requested by an end-user through the cloud.
2) Experiment II: the preparation of a spirometry study requested by an end-user and downloaded from an edge node.
3) Experiment III: spirometry studies and tomography images of 10 patients in an emergency room are sent for a second opinion from a health specialist in a remote geographic location.
4) Experiment IV: the three services were simultaneously executed to process the request of a health application for the results of eight patients.

2) EXPERIMENTAL RESULTS

Fig. 20 depicts the timeline for the Experiment I, where incoming and outcoming continuity schemes were applied to a tomography image of one patient that was requested by an end-user through the cloud. To serve the request, the contents were downloaded by the *blocks* of the services from an edge node. Fig. 20 shows, in the horizontal axis, the *blocks* included in the service. As expected, the *microblocks* of transporting data (*SecUp*) and the incoming of data (*DownSec*) are the tasks consuming the major portion of the response time, whereas the implicit management actions performed by the service only represents a small portion of this response time.

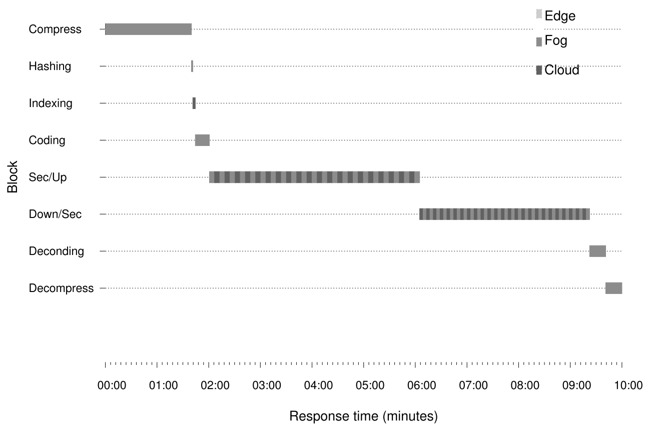


FIGURE 20. Timeline of the continuity schemes of a tomography image requested by an end-user and downloaded from an edge node.

The constant communication between the *blocks* deployed on the fog and those deployed on the cloud is evident.

Fig. 21 shows that the performance of the *blocks* for the Experiment II, when processing spirometry studies is similar to the timeline of the processing tomography images showed in Fig. 20. This means that the major costs in this workflow are produced by data transportation from one environment to another one.

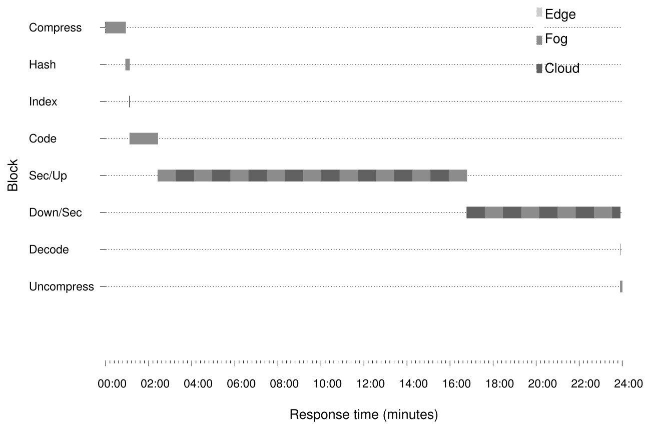


FIGURE 21. Timeline for the outcoming and incoming of a spirometry study requested by an end-user and downloaded from an edge node.

In Experiment III, spirometry studies and tomography images of 10 patients in an emergency room are sent for a second opinion from a health specialist in a remote geographic location. Fig. 22 shows the timeline of the service times for two simultaneous services: one for processing and delivering tomography images and the other for spirometry studies. Both services were configured to manage the contents of 10 patients and deliver these contents to medical staff through health applications in secure and reliable manners.

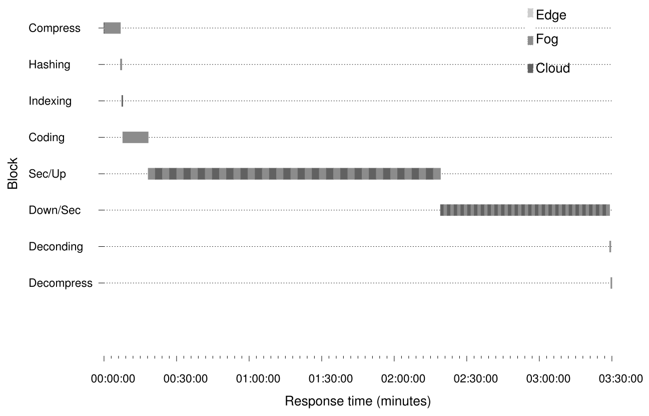


FIGURE 22. Timeline for the outcoming and incoming of health data (spirometry + tomographies) for ten patients.

We recall that the mean size of a spirometry study is 5 GBs, whereas the mean size of an image is 300 MB. In this context, 53 GBs contents were delivered from the fog to the cloud (Amazon) in 2 hours and 38 minutes, to preserve them.

This content is retrieved from the cloud by the health application in another region of Mexico in 1 hour and 35 minutes.

Again, performance is similar to the previous two services (processing images and spirometry studies) showed in Figs. 20 and 21 respectively.

Fig. 23 shows the results of Experiment IV, in which were executed simultaneously to process the request of a health application for the results of eight patients (44 GBs). The expedients were extracted from the emergency room (at the *fog* of the Mexican site), sent to the *cloud* (Amazon) (previously prepared for this transportation) and then downloaded by a health application in a different location *fog* for asking a second opinion.

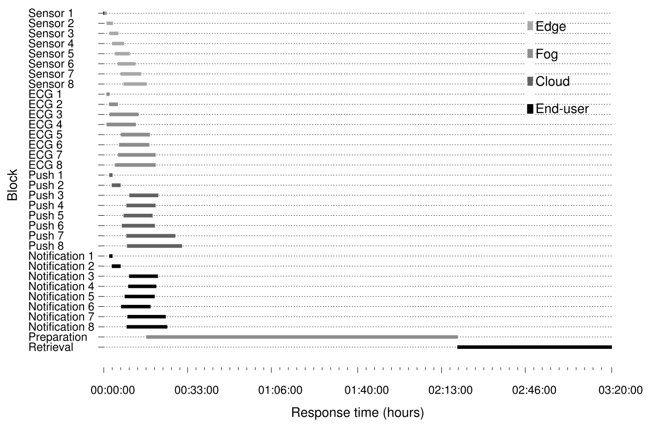


FIGURE 23. Timeline for the acquisition of ECG data and their processing, plus the outcoming continuity scheme of spirometry studies and tomography for eight patients.

The expedients were prepared in 25 minutes (in Mexico), sent to the cloud (Amazon) in 1.75 hours, and delivered to a health application in another location (in Mexico) in 57 minutes. In practice, it is expected that the emergency room may have at least a server to receive data in advance (acting as a *fog*) for temporizing the delays of transportation. Therefore, this experiment represents the worst scenario as all the expedients are requested by one single health application from the cloud not from the hospital servers.

As can be seen, the continuous processing by enforcing the fulfilling of NFRs is feasible in real scenarios, for both reduced and complex scenarios. It is also feasible for observing the policies of health data management laws in terms of access controls as shown in Section IV-E. This means that the contents are not clear at the cloud and even at the fog in some scenarios. Only the producers of the contents (hospitals) and the consumers (health professionals and patients) are enabled to get access in a clear manner. To this end, original contents are encoded and secured in order to only producers and consumers can decode them.

F. EVALUATING THE DATA STORAGE COSTS PRODUCED BY EDGE-FOG-CLOUD PROCESSING STRUCTURES

In this section, are evaluated the storage utilization and the overhead generated by continuous delivery (*CD*), continuous security (*CS*), continuous reliability (*CR*), and continuous efficiency (*CE*) schemes.

1) EXPERIMENTS AND CONFIGURATIONS

The following scenarios were considered in this experimental evaluation:

1) *Scenario 1*: the preparation of tomography images at the fog, which was acquired at the edge, and downloaded by an end-user computer from the cloud.
2) *Scenario 2*: the preparation of spirometry studies at the fog, its preservation in the cloud, and its acquisition from end-user devices.
3) *Scenario 3*: the preparation of the spirometry studies and tomography imagery of eight patients from the edge to an end-user computer.

For each scenario, we evaluated the following three configurations of the continuity provided by the incoming and outcoming continuity schemes:

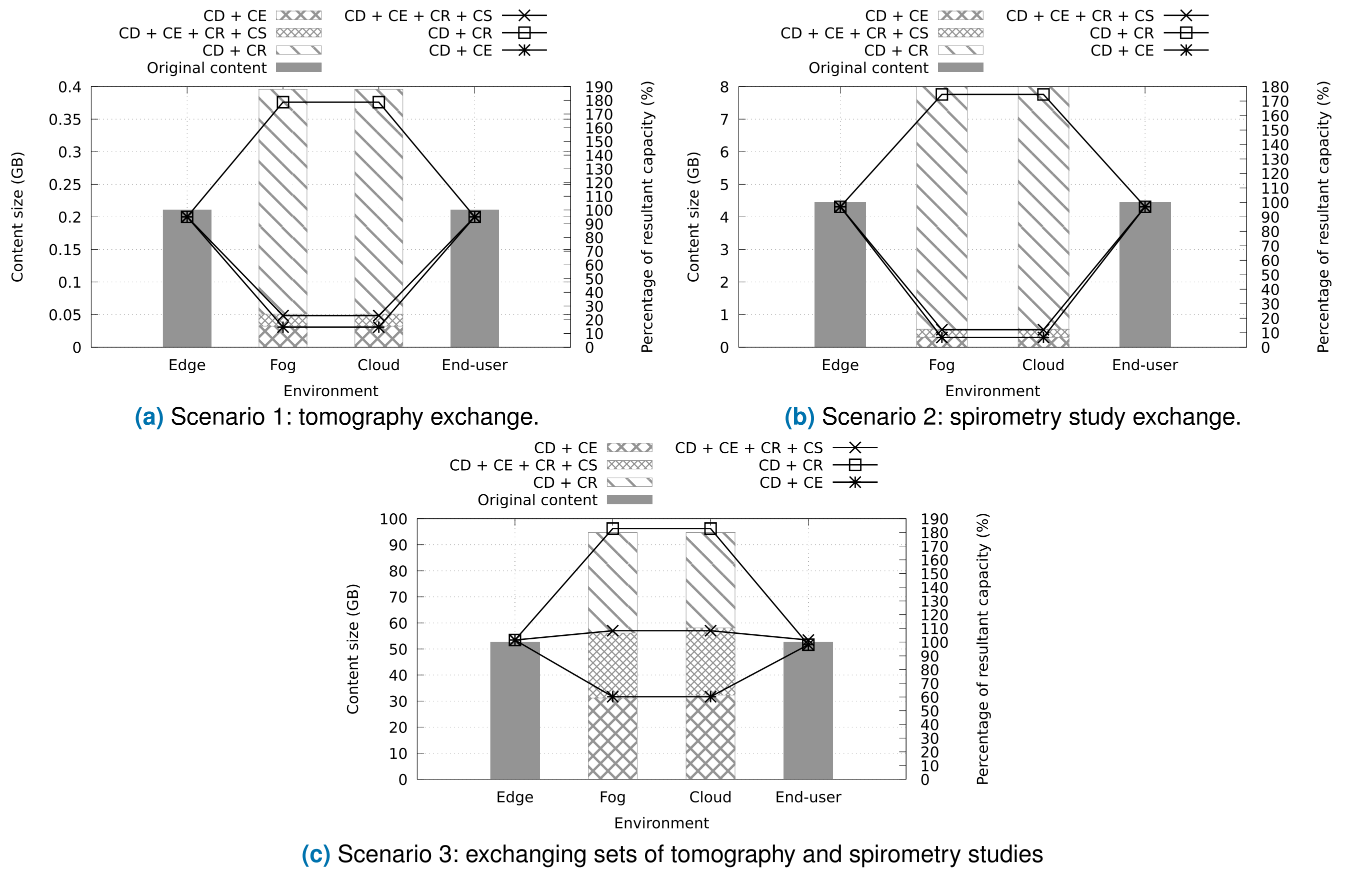
- $CD + CE$: in this configuration, the data are only compressed by a parallel pattern before sending them to the next environment;
- $CD+CR$: in this configuration, the data are only encoded to withstand failures before sending them to the next environment;
- $CD + CE + CR + CS$: in this configuration, the data are prepared to send them to different environments by applying security (*CS*), reliability (*CR*) and cost-efficiency (*CE*).

2) EXPERIMENTAL RESULTS

Fig. 24 shows, left-vertical axis, the storage utilization, and the percentage of a resultant capacity of storage after applying an outcoming scheme to the data (right-vertical axis). The three scenarios considered in this evaluation and previously described are shown in Figs. 24a, 24b, and 24c respectively. As it can be seen in the graphs shown in Fig. 24, the $CD + CR$ represents the configuration producing the highest costs (redundancy is added to the data), whereas the $CD + CE$ produces the lowest costs. The original contents are not transported but encoded (prepared), which is the goal when managing sensitive data in this type of scenario.

The contents acquired from the edge and delivered to the end-users are both exact copies; as a result, the 100% of capacity observed in both edge and end-user environments for all tested scenarios is represented by the filled bar in the plot. This means the configurations producing more than 100% are producing a capacity overhead. See mainly $CD + CR$ in all graphs showed in Fig. 24. In comparison with original data (acquired at the edge and delivered to the end-users) this configuration produced a 88%, 82% and 80% of overhead in Fig. 24a, Fig. 24b and Fig. 24c respectively.

As it can be seen, the $CD + CE + CR + CS$ configuration that applies the three continuity schemes (security, reliability and storage cost-efficiency) produces a quite reduced overhead at the worst scenario (see 6.64% of overhead for this configuration in Fig. 24c in comparison with original data).



(a) Scenario 1: tomography exchange. (b) Scenario 2: spirometry study exchange. (c) Scenario 3: exchanging sets of tomography and spirometry studies

FIGURE 24. Storage utilization by applying the different continuous characteristics to the incoming and outcoming continuity schemes.

Besides, the $CD + CE + CR + CS$ configuration also produces a reduction in the capacity to be transported to the fog and the cloud (see lines for this configuration in the graphs of Fig. 24). Moreover, this configuration also produces an improvement in the resultant capacity when applying these schemes to the processing structures (see bars for these configurations in all graphs presented by Fig. 24). Specifically, Fig. 24a shows that the additional capacity (overhead) created when processing images (tomography) acquired at the edge (204.85 MB), to add only reliability property ($CD+CR$) was of 180.28 MBs. This overhead is produced by the redundancy added to the contents by the IDA algorithm used in CR continuity scheme. In the case of the continuous efficiency property, ($CD + CE$), the size of the contents is reduced to 31.48 MB. When the continuity scheme built by $CD + CE + CR + CS$ configuration by adding security, reliability, and storage/transportation cost-efficiency properties to the contents, the resultant size of the processed image was 49.48 MB. This means no capacity overhead was produced by this continuity scheme.

Similar behavior is observed in Fig. 24b in which was processed a spirometry study by using the three configurations described. The spirometry study has a size of 4.31 GB. $CD + CR$ configuration increases the size of this content to 7.31 GBs. In turn, the size of the data was reduced to 0.29 GB ($\sim$ 298 MB), whereas the resultant capacity produced by $CD + CE + CR + CS$ configuration was 0.53 GB ($\sim$ 536.40 MB).

In the scenarios showed in Fig. 24c, when 10 spirometry and 10 tomographies were prepared by using the three configurations, the initial size of the dataset was 53.44 GB. $CD+CR$ produced an increment of 42,76 GBs (as the resultant capacity was 96.20 GBs). As expected, $CD + CE$ reduced the dataset to 31.66 GB, whereas $CD + CE + CR + CS$ configuration produced almost the same capacity than the original dataset (67.99 MB was the overhead produced by this outcoming scheme).

The results of the experiments showed in Figs. 24a and 24b reveals a significant reduction in the data to be transported to the fog and to the cloud (it only represents a fraction of the original health data) when using the continuous schemes ($CD + CE + CR + CS$) proposed in this paper to prepare the data.

In the worst-case scenario when managing a set of concurrent heterogeneous health data processed by simultaneous services, the costs of assuring the data in terms of continuous security (integrity, confidentiality, and privacy), and continuous reliability (fault tolerance to withstand lost data and cloud outages) was only of 6% (capacity overhead). In this sense, note that in traditional approaches assuring data could represent 67% for codification schemes ($n = 5$, $m/k = 3$) and 200% for replication schemes.

G. RESULTS DISCUSSION: FLEXIBILITY OF THE MODEL AND COMPLEXITY REDUCTION MANAGEMENT

The experimental evaluation has revealed the efficacy and efficiency of the continuity schemes presented in this paper. The continuous delivery (CD) enables the developers to create comprehensive infrastructure-agnostic processing structures for solutions that can be deployed on different environments (any of edge, fog, cloud, or end-users' devices). The flexibility of this type of scheme provides solutions with a dynamic composition of nano/microservices for fulfilling functional requirements.

The continuous security and reliability ($CS + CR$) enable developers to accomplish the strict rules associated with sensitive data, which also are based on the CD continuity scheme to create dataflows for the processing structures deployed in different environments.

The overhead and computational complexity produced by the continuity schemes ($CD + CS + CR$) are issues managed by continuous efficiency (CE) by using two strategies. The first one is focused on reducing the overhead produced by the continuity schemes by implementing parallel patterns, which performs the tasks of the $CD+CS+CR$ schemes in a concurrent manner. This has a direct impact on the service times of the continuity schemes, and thus, in the response times at each continuity stage, which improves the experience of the end-users. The second strategy is focused on reducing the address space by using data compression before executing any task of the continuity schemes, and before sending data from an environment to another one. This significantly improves the performance of continuity schemes, as the tasks performed by these schemes are dealing with less data than without doing this previous process.

This means the continuous management of properties improves the protection of the data that are managed through different environments (fog, cloud, or end-users). Moreover, in the worst case, these tasks are achieved at non-significant cost as, in some cases, even there is no overhead observed when performing these tasks. The beneficial effects of these continuity schemes are produced by two causes: *i)* the compression processes as the first stage in the schemes, which reduces the capacity processed by the next stages and compensates the redundancy added by other stages, and *ii)* the usage of parallel patterns, which reduces the service times of the continuity schemes.

All the experiments described in the previous sections of this paper were performed using a $CD+CE+CR+CS$ configuration because all the scenarios considered the processing of very sensitive data. This configuration seems a framework for organizations to support the continuous processing of sensitive data over heterogeneous and honest-but-curious environments, fulfilling NFRs such as cost-efficiency, security, and reliability.

V. CONCLUSION AND FUTURE WORK

This paper presented the design, development, and implementation of an architectural model based on recursive and modular structures called constructive *blocks*. These *blocks* can be used to create patterns for the edge-fog-cloud computing environment. The secure, reliable, and efficient management of data are tasks performed inside of these structures in an implicit manner. The experimental evaluation revealed the feasibility of using the infrastructure-agnostic edge-fog-cloud processing structures for facing up of the challenge of producing different types of continuity schemes in real scenarios.

The case studies showed the efficacy of applying the following schemes to the management of sensitive big IoT health data:

- *Continuous Delivery* (CD), for creating processing structures that can be deployed on different environments to process sensitive data from IoT to the end-users through different environments (edge, fog, and cloud).
- *Continuous Security* (CS), for producing integrity, confidentiality, and access controls at each stage of the sensitive data processing lifecycle.
- *Continuous Reliability* (CR), for adding fault tolerance to data for withstanding lost data as well as cloud outages and/or fog servers unavailability.
- *Continuous Efficiency* (CE), for improving the delivery and processing of data by applying two techniques: implicit parallelism on the processing tasks, and data compressing to reduce the volume of the data stored.

Advantages of the architectural model over state-of-the-art proposals such as cost-efficiency storage consumption, reduction of data transportation, and improvement of the delivering data to the end-users have been observed in the study cases evaluated in real scenarios. The achieved advantages by this architectural model can be explained as follows: the compression processes performed at the first stage in the schemes reduce the capacity processed by the next stages and compensates the redundancy added by other stages. The usage of parallel patterns reduces the service times of the continuity schemes, and the in-memory storage, in the input/output interfaces, reduces the expensive I/O tasks performed when exchanging data during the stages of the continuity schemes.

Future work will include optimizing the performance of some components of the system, creating new components for different instruments, and extending the architecture to create complex networks of devices and intelligent IoT devices so that we can include the monitoring of multiple IoT devices and/or data sources for several hospitals in the system.

REFERENCES

[1] L. Mainetti, L. Patrono, A. Secco, and I. Sergi, "An IoT-aware AAL system for elderly people," in *Proc. Int. Multidisciplinary Conf. Comput. Energy Sci. (SpliTech)*, Jul. 2016, pp. 1–6.

[2] M. A. Hail and S. Fischer, "IoT for AAL: An architecture via information-centric networking," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, Dec. 2015, pp. 1–6.

[3] P. Rashidi and A. Mihailidis, "A survey on ambient-assisted living tools for older adults," *IEEE J. Biomed. Health Informat.*, vol. 17, no. 3, pp. 579–590, May 2013.

[4] A. Vegesna, M. Tran, M. Angelaccio, and S. Arcona, "Remote patient monitoring via non-invasive digital technologies: A systematic review," *Telemed. e-Health*, vol. 23, no. 1, pp. 3–17, Jan. 2017.

[5] K. N. Griggs, O. Ossipova, C. P. Kohlios, A. N. Baccarini, E. A. Howson, and T. Hayajneh, "Healthcare blockchain system using smart contracts for secure automated remote patient monitoring," *J. Med. Syst.*, vol. 42, no. 7, p. 130, Jul. 2018.

[6] N. Scarpato, A. Pieroni, L. D. Nunzio, and F. Fallucchi, "E-health-IoT universe: A review," *Management*, vol. 21, no. 44, p. 46, 2017.

[7] D. V. Dimitrov, "Medical Internet of Things and big data in healthcare," *Healthcare Inform. Res.*, vol. 22, no. 3, pp. 156–163, 2016.

[8] Y. Yin, Y. Zeng, X. Chen, and Y. Fan, "The Internet of Things in healthcare: An overview," *J. Ind. Inf. Integr.*, vol. 1, pp. 3–13, Mar. 2016.

[9] M. Elhoseny, A. Abdelaziz, A. S. Salama, A. M. Riad, K. Muhammad, and A. K. Sangaiah, "A hybrid model of Internet of Things and cloud computing to manage big data in health services applications," *Future Gener. Comput. Syst.*, vol. 86, pp. 1383–1394, Sep. 2018.

[10] K. Tsiounia, N. Dimitrioglou, D. Kardaras, and S. Barbounaki, "A process modelling and analytic hierarchy process approach to investigate the potential of the IoT in health services," in *World Congress on Medical Physics and Biomedical Engineering*. Singapore: Springer, 2019, pp. 381–386.

[11] Y. Khan, A. E. Ostfeld, C. M. Lochner, A. Pierre, and A. C. Arias, "Monitoring of vital signs with flexible and wearable medical devices," *Adv. Mater.*, vol. 28, no. 22, pp. 4373–4395, Jun. 2016.

[12] H. Wu, Q. Liu, W. Du, C. Li, and G. Shi, "Transparent polymeric strain sensors for monitoring vital signs and beyond," *ACS Appl. Mater. Interface*, vol. 10, no. 4, pp. 3895–3901, Jan. 2018.

[13] J. Liu, Y. Chen, Y. Wang, X. Chen, J. Cheng, and J. Yang, "Monitoring vital signs and postures during sleep using WiFi signals," *IEEE Internet Things J.*, vol. 5, no. 3, pp. 2071–2084, Jun. 2018.

[14] K. Lin, F. Xia, W. Wang, D. Tian, and J. Song, "System design for big data application in emotion-aware healthcare," *IEEE Access*, vol. 4, pp. 6901–6909, 2016.

[15] S. V. B. Peddi, P. Kuhad, A. Yassine, P. Pouladzadeh, S. Shirmohammadi, and A. A. N. Shirehjini, "An intelligent cloud-based data processing broker for mobile e-health multimedia applications," *Future Gener. Comput. Syst.*, vol. 66, pp. 71–86, Jan. 2017.

[16] T. Park, N. Abuzainab, and W. Saad, "Learning how to communicate in the Internet of Things: Finite resources and heterogeneity," *IEEE Access*, vol. 4, pp. 7063–7073, 2016.

[17] S. Verma, Y. Kawamoto, Z. M. Fadlullah, H. Nishiyama, and N. Kato, "A survey on network methodologies for real-time analytics of massive IoT data and open research issues," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1457–1477, 3rd Quart., 2017.

[18] Z. Zhao, G. Min, W. Gao, Y. Wu, H. Duan, and Q. Ni, "Deploying edge computing nodes for large-scale IoT: A diversity aware approach," *IEEE Internet Things J.*, vol. 5, no. 5, pp. 3606–3614, Oct. 2018.

[19] T. N. Gia, M. Jiang, A.-M. Rahmani, T. Westerlund, P. Liljeberg, and H. Tenhunen, "Fog computing in healthcare Internet of Things: A case study on ECG feature extraction," in *Proc. IEEE Int. Conf. Comput. Inf. Technol., Ubiquitous Comput. Commun., Dependable, Autonomic Secure Comput., Pervasive Intell. Comput.*, Oct. 2015, pp. 356–363.

[20] B. Farahani, F. Firouzi, V. Chang, M. Badaroglu, N. Constant, and K. Mankodiya, "Towards fog-driven IoT eHealth: Promises and challenges of IoT in medicine and healthcare," *Future Gener. Comput. Syst.*, vol. 78, pp. 659–676, Jan. 2018.

[21] P. Pace, G. Aloi, R. Gravina, G. Caliciuri, G. Fortino, and A. Liotta, "An edge-based architecture to support efficient applications for healthcare industry 4.0," *IEEE Trans. Ind. Informat.*, vol. 15, no. 1, pp. 481–489, Jan. 2019.

[22] Z. Goli-Malekabadi, M. Sargolzaei-Javan, and M. K. Akbari, "An effective model for store and retrieve big health data in cloud computing," *Comput. Methods Programs Biomed.*, vol. 132, pp. 75–82, Aug. 2016.

[23] S. Huh, S. Cho, and S. Kim, "Managing IoT devices using blockchain platform," in *Proc. 19th Int. Conf. Adv. Commun. Technol. (ICACT)*, Feb. 2017, pp. 464–467.

[24] R. Almadhoun, M. Kadadha, M. Alhemeiri, M. Alshehhi, and K. Salah, "A user authentication scheme of IoT devices using blockchain-enabled fog nodes," in *Proc. IEEE/ACS 15th Int. Conf. Comput. Syst. Appl. (AICCSA)*, Oct. 2018, pp. 1–8.

[25] E. Ahmed, I. Yaqoob, I. A. T. Hashem, I. Khan, A. I. A. Ahmed, M. Imran, and A. V. Vasilakos, "The role of big data analytics in Internet of Things," *Comput. Netw.*, vol. 129, pp. 459–471, Dec. 2017.

[26] W. Shi and S. Dustdar, "The promise of edge computing," *Computer*, vol. 49, no. 5, pp. 78–81, May 2016.

[27] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, Oct. 2016.

[28] Z. Yang, Q. Zhou, L. Lei, K. Zheng, and W. Xiang, "An IoT-cloud based wearable ECG monitoring system for smart healthcare," *J. Med. Syst.*, vol. 40, no. 12, p. 286, Dec. 2016.

[29] K. Hwang and M. Chen, *Big-Data Analytics for Cloud, IoT and Cognitive Computing*. Hoboken, NJ, USA: Wiley, 2017.

[30] M. Wazid, A. K. Das, R. Hussain, G. Succi, and J. J. P. C. Rodrigues, "Authentication in cloud-driven IoT-based big data environment: Survey and outlook," *J. Syst. Archit.*, vol. 97, pp. 185–196, Aug. 2019.

[31] H. S. Narman, M. S. Hossain, M. Atiquzzaman, and H. Shen, "Scheduling Internet of Things applications in cloud computing," *Ann. Telecommun.*, vol. 72, nos. 1–2, pp. 79–93, Feb. 2017.

[32] T. J.-J. Li, Y. Li, F. Chen, and B. A. Myers, "Programming IoT devices by demonstration using mobile Apps," in *Proc. Int. Symp. End User Develop.* Cham, Switzerland: Springer, 2017, pp. 3–17.

[33] J. Coutaz and J. L. Crowley, "A first-person experience with end-user development for smart homes," *IEEE Pervas. Comput.*, vol. 15, no. 2, pp. 26–39, Apr. 2016.

[34] H. Cao, M. Wachowicz, C. Renso, and E. Carlini, "Analytics everywhere: Generating insights from the Internet of Things," *IEEE Access*, vol. 7, pp. 71749–71769, 2019.

[35] C. H. Mier and V. T. Delgadillo, "Regulación del acceso al expediente clínico con fines de investigación en México," *Revista CONAMED*, vol. 22, no. 1, pp. 27–31, 2018.

[36] M. Phillips, "International data-sharing norms: From the OECD to the general data protection regulation (GDPR)," *Hum. Genet.*, vol. 137, no. 8, pp. 575–582, Aug. 2018.

[37] K. Bhushan and B. B. Gupta, "Security challenges in cloud computing: State-of-art," *Int. J. Big Data Intell.*, vol. 4, no. 2, pp. 81–107, 2017.

[38] R. French-Baidoo, D. Asamoah, and S. O. Oppong, "Achieving confidentiality in electronic health records using cloud systems," *Int. J. Comput. Netw. Inf. Secur.*, vol. 10, no. 1, p. 18, 2018.

[39] A. Singh and K. Chatterjee, "Cloud security issues and challenges: A survey," *J. Netw. Comput. Appl.*, vol. 79, pp. 88–115, Feb. 2017.

[40] I. Stojmenovic and S. Wen, "The fog computing paradigm: Scenarios and security issues," in *Proc. Federated Conf. Comput. Sci. Inf. Syst.*, Sep. 2014, pp. 1–8.

[41] M. Mukherjee, R. Matam, L. Shu, L. Maglaras, M. A. Ferrag, N. Choudhury, and V. Kumar, "Security and privacy in fog computing: Challenges," *IEEE Access*, vol. 5, pp. 19293–19304, 2017.

[42] M. Frustaci, P. Pace, G. Aloi, and G. Fortino, "Evaluating critical security issues of the IoT world: Present and future challenges," *IEEE Internet Things J.*, vol. 5, no. 4, pp. 2483–2495, Aug. 2018.

[43] N. Vurukonda and B. T. Rao, "A study on data storage security issues in cloud computing," *Procedia Comput. Sci.*, vol. 92, pp. 128–135, Jan. 2016.

[44] H. S. Gunawi, M. Hao, R. O. Suminto, A. Laksono, A. D. Satria, J. Adityatama, and K. J. Eliazar, "Why does the cloud stop computing?: lessons from hundreds of service outages," in *Proc. 7th ACM Symp. Cloud Comput.*, Oct. 2016, pp. 1–16.

[45] A. Bala and I. Chana, "Fault tolerance-challenges, techniques and implementation in cloud computing," *Int. J. Comput. Sci. Issues*, vol. 9, no. 1, p. 288, 2012.

[46] A. V. Dastjerdi and R. Buyya, "Fog computing: Helping the Internet of Things realize its potential," *Computer*, vol. 49, no. 8, pp. 112–116, Aug. 2016.

[47] E. N. Power, "Understanding the cost of data center downtime: An analysis of the financial impact on infrastructure vulnerability," Emerson Netw. Power, Columbus, OH, USA, White Paper SL-24661 R05-11, 2011.

[48] TI Process Institute. (Jun. 2019). *The Real Cost of Business Interruption*. Accessed: Jul. 15, 2019. [Online]. Available: http://calyxit.com/the-real-cost-of-business-interruption/

[49] Ponemon Institute' Research Report. (2011). *Calculating the Cost of Data Center Outages*. Accessed: Sep. 19, 2019. [Online]. Available: https://airandpowersolutions.com/wp-content/uploads/2015/09/Calculating-the-%Cost-of-Data-Center-Outages-Ponemon-Institute-White-Paper-R0211-SL-24659.pdf

[50] M. Li, C. Qin, and P. P. Lee, "Cdstore: Toward reliable, secure, and cost-efficient cloud storage via convergent dispersal," in *Proc. USENIX Annu. Tech. Conf. (USENIX ATC)*, 2015, pp. 111–124.

[51] A. Reichman, *File Storage Costs Less in the Cloud Than in-House*. Cambridge, MA, USA: Forrester Research, 2011.

[52] Y. Demchenko, P. Grosso, C. de Laat, and P. Membrey, "Addressing big data issues in scientific data infrastructure," in *Proc. Int. Conf. Collaboration Technol. Syst. (CTS)*, May 2013, pp. 48–55.

[53] F. L. F. Almeida and C. Calistru, "The main challenges and issues of big data management," *Int. J. Res. Stud. Comput.*, vol. 2, no. 1, pp. 11–20, Apr. 2013.

[54] M. Fröhlich, M. Mutschler, M. Caspers, U. Nienaber, V. Jäcker, A. Driessen, B. Bouillon, and M. Maegele, "Trauma-induced coagulopathy upon emergency room arrival: Still a significant problem despite increased awareness and management?" *Eur. J. Trauma Emergency Surg.*, vol. 45, no. 1, pp. 115–124, Feb. 2019.

[55] D. Laborde, "System, client device, server and method for providing a cross-facility patient data management and reporting platform," U.S. Patent 10 354 750, Jul. 16, 2019.

[56] S. L. Albuquerque and P. R. L. Gondim, "Security in cloud-computing-based mobile health," *IT Prof.*, vol. 18, no. 3, pp. 37–44, May 2016.

[57] A. Mxoli, M. Gerber, and N. Mostert-Phipps, "Information security risk measures for cloud-based personal health records," in *Proc. Int. Conf. Inf. Soc. (i-Soc.)*, Nov. 2014, pp. 208–216.

[58] J. J. M. Seddon and W. L. Currie, "Cloud computing and trans-border health data: Unpacking U.S. and EU healthcare regulation and compliance," *Health Policy Technol.*, vol. 2, no. 4, pp. 229–241, Dec. 2013.

[59] L. J. Sotto, B. C. Treacy, and M. L. McLellan, "Privacy and data security risks in cloud computing," *World Commun. Regulation Report*, vol. 5, no. 2, p. 38, 2010.

[60] J. Zhou, Z. Cao, X. Dong, and A. V. Vasilakos, "Security and privacy for cloud-based IoT: Challenges," *IEEE Commun. Mag.*, vol. 55, no. 1, pp. 26–33, Jan. 2017.

[61] I. Lee and K. Lee, "The Internet of Things (IoT): Applications, investments, and challenges for enterprises," *Bus. Horizons*, vol. 58, no. 4, pp. 431–440, Jul. 2015.

[62] M. A. Khan and K. Salah, "IoT security: Review, blockchain solutions, and open challenges," *Future Gener. Comput. Syst.*, vol. 82, pp. 395–411, May 2018.

[63] Z. B. Celik, E. Fernandes, E. Pauley, G. Tan, and P. McDaniel, "Program analysis of commodity IoT applications for security and privacy: Challenges and opportunities," *ACM Comput. Surv.*, vol. 52, no. 4, pp. 1–30, Sep. 2019.

[64] D. Bernstein, "Containers and cloud: From LXC to docker to kubernetes," *IEEE Cloud Comput.*, vol. 1, no. 3, pp. 81–84, Sep. 2014.

[65] C. de Alfonso, A. Calatrava, and G. Moltó, "Container-based virtual elastic clusters," *J. Syst. Softw.*, vol. 127, pp. 1–11, May 2017.

[66] 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*. Cham, Switzerland: Springer, 2017, pp. 195–216.

[67] Sutrisno, F. Panduwinata, and P. Yugopuspito, "Nanoservices as generalization services in service-oriented architecture," in *Proc. Int. Conf. Soft Comput., Intell. Syst. Inf. Technol. (ICSIIT)*, Sep. 2017, pp. 131–137.

[68] J. Islam, E. Harjula, T. Kumar, P. Karhula, and M. Ylianttila, "Docker enabled virtualized nanoservices for local IoT edge networks," in *Proc. IEEE Conf. Standards Commun. Netw. (CSCN)*, Oct. 2019, pp. 1–7.

[69] K. Lee, A. Agrawal, and A. Choudhary, "Real-time disease surveillance using Twitter data: Demonstration on flu and cancer," in *Proc. 19th ACM SIGKDD Int. Conf. Knowl. Dscovery Dta Mning KDD*, 2013, pp. 1474–1477.

[70] R. A. Neher and T. Bedford, "Real-time analysis and visualization of pathogen sequence data," *J. Clin. Microbiol.*, vol. 56, no. 11, pp. 1–9, Aug. 2018.

[71] J. B. Seal, C. Stewart, J. McGee, T. Nguyen, D. Sonnier, R. Milani, G. Loss, and K. Sarkar, "Image processing workflow for virtual and augmented reality platforms in liver surgery," *HPB*, vol. 21, p. S180, Mar. 2019.

[72] P. Ohnemus, A. Naef, L. Jacobs, and D. Leason, "Automated health data acquisition, processing and communication system," U.S. Patent 8 706 530, Apr. 22, 2014.

[73] L. Catarinucci, D. de Donno, L. Mainetti, L. Palano, L. Patrono, M. L. Stefanizzi, and L. Tarricone, "An IoT-aware architecture for smart healthcare systems," *IEEE Internet Things J.*, vol. 2, no. 6, pp. 515–526, Dec. 2015.

[74] E. Choi, M. T. Bahadori, A. Schuetz, W. F. Stewart, and J. Sun, "Doctor ai: Predicting clinical events via recurrent neural networks," in *Proc. Mach. Learn. Healthcare Conf.*, 2016, pp. 301–318.

[75] G. Alex, B. Varghese, J. G. Jose, and A. Abraham, "A modern health care system using IoT and Android," *Int. J. Comput. Sci. Eng.*, vol. 8, no. 4, 2016.

[76] L. Zhang, C. Wu, T. Yoshinaga, X. Chen, T. Murase, and Y. Ji, "Multihop data delivery virtualization for green decentralized IoT," *Wireless Commun. Mobile Comput.*, vol. 2017, pp. 1–9, Dec. 2017.

[77] P. L. R. Chze and K. S. Leong, "A secure multi-hop routing for IoT communication," in *Proc. IEEE World Forum Internet Things (WF-IoT)*, Mar. 2014, pp. 428–432.

[78] A. Shatnawi, M. Orrù, M. Mobilio, O. Riganelli, and L. Mariani, "Cloud-health: A model-driven approach to watch the health of cloud services," in *Proc. 1st Int. Workshop Softw. Health SoHeal*, May 2018, pp. 40–47.

[79] J. R. Vest, S. J. Grannis, D. P. Haut, P. K. Halverson, and N. Menachemi, "Using structured and unstructured data to identify patients' need for services that address the social determinants of health," *Int. J. Med. Informat.*, vol. 107, pp. 101–106, Nov. 2017.

[80] L. Jiang, L. Da Xu, H. Cai, Z. Jiang, F. Bu, and B. Xu, "An IoT-oriented data storage framework in cloud computing platform," *IEEE Trans. Ind. Informat.*, vol. 10, no. 2, pp. 1443–1451, May 2014.

[81] V. N. Kumar and P. Shindgikar, *Modern Big Data Processing With Hadoop: Expert Techniques for Architecting End-to-End Big Data Solutions to Get Valuable Insights*. Birmingham, U.K.: Packt Publishing, 2018.

[82] L. R. Nair, S. D. Shetty, and S. D. Shetty, "Applying spark based machine learning model on streaming big data for health status prediction," *Comput. Electr. Eng.*, vol. 65, pp. 393–399, Jan. 2018.

[83] J. Opara-Martins, R. Sahandi, and F. Tian, "Critical review of vendor lock-in and its impact on adoption of cloud computing," in *Proc. Int. Conf. Inf. Soc. (i-Soci.*, Nov. 2014, pp. 92–97.

[84] J. Opara-Martins, R. Sahandi, and F. Tian, "Critical analysis of vendor lock-in and its impact on cloud computing migration: A business perspective," *J. Cloud Comput.*, vol. 5, no. 1, p. 4, Dec. 2016.

[85] D. Carrizales, D. D. Sánchez-Gallegos, H. Reyes, J. Gonzalez-Compean, M. Morales-Sandoval, J. Carretero, and A. Galaviz-Mosqueda, "A data preparation approach for cloud storage based on containerized parallel patterns," in *Proc. Int. Conf. Internet Distrib. Comput. Syst.* Cham, Switzerland: Springer, 2019, pp. 478–490.

[86] J. L. Gonzalez-Compean, V. Sosa-Sosa, A. Diaz-Perez, J. Carretero, and J. Yanez-Sierra, "Sacbe: A building block approach for constructing efficient and flexible end-to-end cloud storage," *J. Syst. Softw.*, vol. 135, pp. 143–156, Jan. 2018.

[87] S. S. Samant, M. Baruwal Chhetri, Q. Bao Vo, R. Kowalczyk, and S. Nepal, "Towards end-to-end QoS and cost-aware resource scaling in cloud-based IoT data processing pipelines," in *Proc. IEEE Int. Conf. Services Comput. (SCC)*, Jul. 2018, pp. 287–290.

[88] X. A. Wang, J. Ma, F. Xhafa, M. Zhang, and X. Luo, "Cost-effective secure E-health cloud system using identity based cryptographic techniques," *Future Gener. Comput. Syst.*, vol. 67, pp. 242–254, Feb. 2017.

[89] R. Kalaiprasath, R. Elankavi, and R. Udayakumar, "Cloud security and compliance—A semantic approach in end to end security," *Int. J. Smart Sens. Intell. Syst.*, vol. 10, no. 4, pp. 482–494, 2017.

[90] T. Pisello and B. Quirk, "How to quantify downtime," *Netw. World*, vol. 5, p. 41, Jan. 2004.

[91] M. O. Rabin, "The information dispersal algorithm and its applications," in *Sequences*. New York, NY, USA: Springer, 1990, pp. 406–419.

[92] S. Kan and J. Dworak, "IJTAG integrity checking with chained hashing," in *Proc. IEEE Int. Test Conf. (ITC)*, Oct. 2018, pp. 1–10.

[93] B. Liu, X. L. Yu, S. Chen, X. Xu, and L. Zhu, "Blockchain based data integrity service framework for IoT data," in *Proc. IEEE Int. Conf. Web Services (ICWS)*, Jun. 2017, pp. 468–475.

[94] S. R. Moosavi, T. N. Gia, E. Nigussie, A. M. Rahmani, S. Virtanen, H. Tenhunen, and J. Isoaho, "End-to-end security scheme for mobility enabled healthcare Internet of Things," *Future Gener. Comput. Syst.*, vol. 64, pp. 108–124, Nov. 2016.

[95] J. L. Gonzalez-Compean, O. Telles, I. Lopez-Arevalo, M. Morales-Sandoval, V. J. Sosa-Sosa, and J. Carretero, "A policy-based containerized filter for secure information sharing in organizational environments," *Future Gener. Comput. Syst.*, vol. 95, pp. 430–444, Jun. 2019.

[96] J. Ren, H. Guo, C. Xu, and Y. Zhang, ''Serving at the edge: A scalable IoT architecture based on transparent computing,'' *IEEE Netw.*, vol. 31, no. 5, pp. 96–105, Aug. 2017.

[97] T. Muhammed, R. Mehmood, A. Albeshri, and I. Katib, ''Ubehealth: A personalized ubiquitous cloud and edge-enabled networked healthcare system for smart cities,'' *IEEE Access*, vol. 6, pp. 32258–32285, 2018.

[98] M. Chen, W. Li, Y. Hao, Y. Qian, and I. Humar, ''Edge cognitive computing based smart healthcare system,'' *Future Gener. Comput. Syst.*, vol. 86, pp. 403–411, Sep. 2018.

[99] M. Al-Khafajiy, L. Webster, T. Baker, and A. Waraich, ''Towards fog driven iot healthcare: Challenges and framework of fog computing in healthcare,'' in *Proc. 2nd Int. Conf. Future Netw. Distrib. Syst.*, 2018, pp. 1–7.

[100] M. Al-khafajiy, T. Baker, C. Chalmers, M. Asim, H. Kolivand, M. Fahim, and A. Waraich, ''Remote health monitoring of elderly through wearable sensors,'' *Multimedia Tools Appl.*, vol. 78, no. 17, pp. 24681–24706, Sep. 2019.

[101] S. M. Babu, A. J. Lakshmi, and B. T. Rao, ''A study on cloud based Internet of Things: CloudIoT,'' in *Proc. Global Conf. Commun. Technol. (GCCT)*, Apr. 2015, pp. 60–65.

[102] N. C. Taher, I. Mallat, N. Agoulmine, and N. El-Mawass, ''An IoT-cloud based solution for real-time and batch processing of big data: Application in healthcare,'' in *Proc. 3rd Int. Conf. Bio-Eng. Smart Technol. (BioSMART)*, Apr. 2019, pp. 1–8.

[103] M. Morales-Sandoval, J. L. Gonzalez-Compean, A. Diaz-Perez, and V. J. Sosa-Sosa, ''A pairing-based cryptographic approach for data security in the cloud,'' *Int. J. Inf. Secur.*, vol. 17, no. 4, pp. 441–461, Aug. 2018.

[104] J. Zhang and Z. Zhang, ''Secure and efficient data-sharing in clouds,'' *Concurrency Comput., Pract. Exper.*, vol. 27, no. 8, pp. 2125–2143, Jun. 2015.

[105] B. Mao, S. Wu, and H. Jiang, ''Improving storage availability in cloud-of-clouds with hybrid redundant data distribution,'' in *Proc. IEEE Int. Parallel Distrib. Process. Symp.*, May 2015, pp. 633–642.

[106] H. Xiong, X. Zhang, W. Zhu, and D. Yao, ''Cloudseal: End-to-end content protection in cloud-based storage and delivery services,'' in *SecureComm*. Berlin, Germany: Springer, 2011, pp. 491–500.

[107] Y. Yu, M. H. Au, G. Ateniese, X. Huang, W. Susilo, Y. Dai, and G. Min, ''Identity-based remote data integrity checking with perfect data privacy preserving for cloud storage,'' *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 4, pp. 767–778, Apr. 2017.

[108] R. Montella, D. Kelly, W. Xiong, A. Brizius, J. Elliott, R. Madduri, K. Maheshwari, C. Porter, P. Vilter, M. Wilde, M. Zhang, and I. Foster, ''FACE-IT: A science gateway for food security research,'' *Concurrency Comput., Pract. Exper.*, vol. 27, no. 16, pp. 4423–4436, Nov. 2015.

[109] J. Goecks, A. Nekrutenko, J. Taylor, and T. G. Team, ''Galaxy: A comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences,'' *Genome Biol.*, vol. 11, no. 8, p. R86, 2010.

[110] Y. Babuji, K. Chard, I. Foster, S. Daniel Katz, M. Wilde, A. Woodard, and J. Wozniak, ''Parsl: Scalable parallel scripting in python,'' in *Proc. 10th Int. Workshop Sci. Gateways*, 2018, pp. 1–6.

[111] C. Zheng and D. Thain, ''Integrating containers into workflows: A case study using makeflow, work queue, and docker,'' in *Proc. 8th Int. Workshop Virtualization Technol. Distrib. Comput. VTDC*, 2015, pp. 31–38.

[112] Z. Wen, J. Cala, P. Watson, and A. Romanovsky, ''Cost effective, reliable and secure workflow deployment over federated clouds,'' *IEEE Trans. Services Comput.*, vol. 10, no. 6, pp. 929–941, Nov. 2017.

[113] K. Maheshwari, E.-S. Jung, J. Meng, V. Morozov, V. Vishwanath, and R. Kettimuthu, ''Workflow performance improvement using model-based scheduling over multiple clusters and clouds,'' *Future Gener. Comput. Syst.*, vol. 54, pp. 206–218, Jan. 2016.

[114] G. L. Stavrinides, F. R. Duro, H. D. Karatza, J. G. Blas, and J. Carretero, ''Different aspects of workflow scheduling in large-scale distributed systems,'' *Simul. Model. Pract. Theory*, vol. 70, pp. 120–134, Jan. 2017.

[115] M. Gabbrielli, S. Giallorenzo, C. Guidi, J. Mauro, and F. Montesi, ''Self-reconfiguring microservices,'' in *Theory and Practice of Formal Methods*. Cham, Switzerland: Springer, 2016, pp. 194–210.

[116] E. Afgan, V. Jalili, N. Goonasekera, J. Taylor, and J. Goecks, ''Federated galaxy: Biomedical computing at the frontier,'' in *Proc. IEEE 11th Int. Conf. Cloud Comput. (CLOUD)*, Jul. 2018, pp. 871–874.

[117] H. Visti, T. Kiss, G. Terstyanszky, G. Gesmier, and S. Winter, ''Micado—Towards a microservice-based cloud application-level dynamic orchestrator,'' in *Proc. 8th Int. Workshop Sci. Gateways, IWSG*, 2017, pp. 1–7.

[118] F. Lordan, E. Tejedor, J. Ejarque, R. Rafanell, J. Álvarez, F. Marozzo, D. Lezzi, R. Sirvent, D. Talia, and R. M. Badia, ''ServiceSs: An interoperable programming framework for the cloud,'' *J. Grid Comput.*, vol. 12, no. 1, pp. 67–91, Mar. 2014.

[119] E. Tejedor, Y. Becerra, G. Alomar, A. Queralt, R. M. Badia, J. Torres, T. Cortes, and J. Labarta, ''PyCOMPSs: Parallel computational workflows in Python,'' *Int. J. High Perform. Comput. Appl.*, vol. 31, no. 1, pp. 66–82, Jan. 2017.

[120] M. Albrecht, P. Donnelly, P. Bui, and D. Thain, ''Makeflow: A portable abstraction for data intensive computing on clusters, clouds, and grids,'' in *Proc. 1st ACM SIGMOD Workshop Scalable Workflow Execution Engines Technol. SWEET*, 2012, pp. 1–13.

[121] Y. Babuji, A. Woodard, Z. Li, D. S. Katz, B. Clifford, R. Kumar, L. Lacinski, R. Chard, J. M. Wozniak, I. Foster, M. Wilde, and K. Chard, ''Parsl: Pervasive parallel programming in Python,'' in *Proc. 28th Int. Symp. High-Perform. Parallel Distrib. Comput.*, Jun. 2019, pp. 25–36.

[122] N. Garg, *Apache Kafka*. Birmingham, U.K.: Packt Publishing, 2013.

[123] S. Mathew and J. Varia, ''Overview of Amazon Web services,'' Amazon, Seattle, WA, USA, White Papers, 2014. [Online]. Available: http://cabibbo.dia.uniroma3.it/asw-2014-2015/altrui/AWS_Overview.pdf

[124] D. D. Sanchez-Gallegos, D. Di Luccio, J. L. Gonzalez-Compean, and R. Montella, ''Internet of Tings orchestration using DagOn* workflow engine,'' in *Proc. IEEE 5th World Forum Internet Things (WF-IoT)*, Apr. 2019, pp. 95–100.

[125] H. C. H. Chen and P. P. C. Lee, ''Enabling data integrity protection in Regenerating-Coding-Based cloud storage: Theory and implementation,'' *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 2, pp. 407–416, Feb. 2014.

[126] J. Ajayakumar, A. J. Curtis, and J. Curtis, ''Addressing the data guardian and geospatial scientist collaborator dilemma: How to share health records for spatial analysis while maintaining patient confidentiality,'' *Int. J. Health Geographics*, vol. 18, no. 1, pp. 1–12, Dec. 2019.

[127] O. Ali and A. Ouda, ''A classification module in data masking framework for business intelligence platform in healthcare,'' in *Proc. IEEE 7th Annu. Inf. Technol., Electron. Mobile Commun. Conf. (IEMCON)*, Oct. 2016, pp. 1–8.

[128] S. Yu, C. Wang, K. Ren, and W. Lou, ''Achieving secure, scalable, and fine-grained data access control in cloud computing,'' in *Proc. IEEE INFOCOM*, Mar. 2010, pp. 1–9.

[129] P. Vassiliadis, ''A survey of extract–transform–load technology,'' *Int. J. Data Warehousing Mining*, vol. 5, no. 3, pp. 1–27, 2009.

[130] J. L. Ortega-Arjona, *Patterns for Parallel Software Design*, 1st ed. Hoboken, NJ, USA: Wiley, 2010.

[131] H. G. Reyes-Anastacio, J. L. Gonzalez-Compean, M. Morales-Sandoval, and J. Carretero, ''A data integrity verification service for cloud storage based on building blocks,'' in *Proc. 8th Int. Conf. Comput. Sci. Inf. Technol. (CSIT)*, Jul. 2018, pp. 201–206.

[132] S. Pu, W. Shi, J. Xu, and A. Nedic, ''Push-pull gradient methods for distributed optimization in networks,'' *IEEE Trans. Autom. Control*, early access, Feb. 10, 2020, doi: 10.1109/TAC.2020.2972824.

[133] K. Zheng, J. Westbrook, T. G. Kannampallil, and V. L. Patel, ''Clinical workflow in the health it era,'' in *Cognitive Informatics*. Cham, Switzerland: Springer, 2019, pp. 3–7.

[134] F. de Asis Lopez Fuentes, J. M. Almanza, R. Marcelin-Jimenez, and B. Velazquez-Mendez, ''Efficient content distribution and storage P2P system based on information dispersal,'' in *Proc. 6th Int. Conf. Control, Decis. Inf. Technol. (CoDIT)*, Apr. 2019, pp. 1604–1609.

[135] M. O. Rabin, ''Efficient dispersal of information for security, load balancing, and fault tolerance,'' *J. ACM (JACM)*, vol. 36, no. 2, pp. 335–348, Apr. 1989.

[136] J. Yanez-Sierra, A. Diaz-Perez, V. Sosa-Sosa, and J. L. Gonzalez, ''Towards secure and dependable cloud storage based on user-defined workflows,'' in *Proc. IEEE 2nd Int. Conf. Cyber Secur. Cloud Comput.*, Nov. 2015, pp. 405–410.

[137] J. Bethencourt, A. Sahai, and B. Waters, ''Ciphertext-policy attribute-based encryption,'' in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2007, pp. 321–334.

[138] S. Pérez, D. Rotondi, D. Pedone, L. Straniero, M. J. N. Núñez, and F. Gigante, ''Towards the CP-ABE application for privacy-preserving secure data sharing in IoT contexts,'' in *Proc. Int. Conf. Innov. Mobile Internet Services Ubiquitous Comput.*, Springer, 2017, pp. 917–926

[139] S. M. R. Islam, D. Kwak, M. H. Kabir, M. Hossain, and K.-S. Kwak, ''The Internet of Things for health care: A comprehensive survey,'' *IEEE Access*, vol. 3, pp. 678–708, 2015.

[140] J. Reyes-Garcia, H. Galeana-Zapien, A. Galaviz-Mosqueda, and C. Torres-Huitzil, "Evaluation of the impact of data uncertainty on the prediction of physiological patient deterioration," *IEEE Access*, vol. 6, pp. 38595–38606, 2018.

[141] M. M. Baig, H. Gholamhosseini, and M. J. Connolly, "A comprehensive survey of wearable and wireless ECG monitoring systems for older adults," *Med. Biol. Eng. Comput.*, vol. 51, no. 5, pp. 485–495, May 2013.

[142] A. Maach, J. el Alami, and E. H. el Mazoudi, "A fog-driven IoT e-Health framework to monitor and control asthma exacerbation," in *Proc. Int. Conf. Wireless Netw. Mobile Commun. (WINCOM)*, Oct. 2019, pp. 1–6.

[143] J. Clemente, M. Valero, J. Mohammadpour, X. Li, and W. Song, "Fog computing middleware for distributed cooperative data analytics," in *Proc. IEEE Fog World Congr. (FWC)*, Oct. 2017, pp. 1–6.

[144] G. Bianchi, "Performance analysis of the IEEE 802.11 distributed coordination function," *IEEE J. Sel. Areas Commun.*, vol. 18, no. 3, pp. 535–547, Mar. 2000.

[145] M. J. Luczak and C. McDiarmid, "On the power of two choices: Balls and bins in continuous time," *Ann. Appl. Probab.*, vol. 15, no. 3, pp. 1733–1764, Aug. 2005.

[146] P. Morales-Ferreira, M. Santiago-Duran, C. Gaytan-Diaz, J. L. Gonzalez-Compean, V. J. Sosa-Sosa, and I. Lopez-Arevalo, "A data distribution service for cloud and containerized storage based on information dispersal," in *Proc. IEEE Symp. Service-Oriented Syst. Eng. (SOSE)*, Mar. 2018, pp. 86–95.

[147] M. Sznajder and M. Lukowska, "Python online and offline ECG QRS detector based on the pan-tomkins algorithm," Jagiellonian Univ., Kraków, Poland, Tech. Rep. 10.5281/zenodo.826614, May 2017.

[148] V. John and X. Liu, "A survey of distributed message broker queues," 2017, *arXiv:1704.00411*. [Online]. Available: http://arxiv.org/abs/1704.00411

[149] T. R. Mayer, L. Brunie, D. Coquil, and H. Kosch, "On reliability in publish/subscribe systems: A survey," *Int. J. Parallel, Emergent Distrib. Syst.*, vol. 27, no. 5, pp. 369–386, 2012.

[150] V. Soleimani, M. Mirmehdi, D. Damen, M. Camplani, S. Hannuna, C. Sharp, and J. Dodd, "Depth-based whole body photoplethysmography in remote pulmonary function testing," *IEEE Trans. Biomed. Eng.*, vol. 65, no. 6, pp. 1421–1431, Jun. 2018.

[151] P. Mildenberger, M. Eichelberg, and E. Martin, "Introduction to the DICOM standard," *Eur. Radiol.*, vol. 12, no. 4, pp. 920–927, Apr. 2002.

[152] B. Albertina, M. Watson, C. Holback, R. Jarosz, S. Kirk, Y. Lee, and J. Lemmerman, "Radiology Data from The Cancer Genome Atlas Lung Adenocarcinoma [TCGA-LUAD] collection," *Cancer Imag. Arch.*, 2016, doi: 10.7937/K9/TCIA.2016.JGNIHEP5.

[153] R. Montella, D. Di Luccio, and S. Kosta, "DagOn*: Executing direct acyclic graphs as parallel jobs on anything," in *Proc. IEEE/ACM Workflows Support Large-Scale Sci. (WORKS)*, Nov. 2018, pp. 64–73.

[154] M. Anwar, A. Gill, and G. Beydoun, "A review of information privacy laws and standards for secure digital ecosystems," in *Proc. 29th Australas. Conf. Inf. Syst.*, Jan. 2018, pp. 1–12.

[155] C. de diputados de México, "Ley general de protección de datos personales en posesión de sujetos obligados," Cámara de Diputados del H. Congreso de la Unión, Mexico City, Mexico, Tech. Rep. DOF 26-01-2017, 2017.

DANTE D. SÁNCHEZ-GALLEGOS received the B.E. degree in IT engineering from the Polytechnic University of Victoria, Mexico, in 2016, and the master's degree in sciences from the Cinvestav Tamaulipas, Mexico, in 2019, where he is currently pursuing the Ph.D. degree. His research interests include processing workflows, distributed systems, and cloud computing.

ALEJANDRO GALAVIZ-MOSQUEDA received the M.Sc. degree in computer science from the University of Colima, Mexico, in 2006, and the Ph.D. degree from the CICESE Research Center, Mexico, in 2013. He is currently a Researcher with the CONACYT-CICESE, Unidad Monterrey, Mexico. His main research interests include mobile and wireless networks for intelligent transport systems and mhealth.

J. L. GONZALEZ-COMPEAN received the Ph.D. degree in computer architecture from the UPC Universitat Politecnica de Catalunya, Barcelona, in 2009. He was a Visiting Professor with the Universidad Carlos III de Madrid, Spain, and a Researcher with the Cinvestav, Mexico. His research lines are cloud-based storage systems, linguistic archival systems, and federated storage networks. His research interests include design of fault-tolerant, adaptability and availability strategies, task scheduling, and storage virtualization.

SALVADOR VILLARREAL-REYES (Member, IEEE) received the B.Sc. degree (Eng.) in electronics engineering from the Durango Institute of Technology, Durango, Mexico, in 1998, the M.Sc. degree in electronics engineering (telecommunications) from the Monterrey Institute of Technology and Higher Education, Monterrey, Mexico, in 2001, and the Ph.D. degree in electrical and electronics engineering from Loughborough University, U.K., in 2007. He is currently a Titular Researcher with the CICESE Research Center, Ensenada, Mexico. His research interests include ad-hoc networks, vehicular ad-hoc networks, flying ad-hoc networks, the IoT, the Internet of Medical Things (m-IoT), e-health systems, m-health systems, wireless sensor networks, digital signal processing, embedded systems, and so on.

ALDO E. PEREZ-RAMOS received the B.Sc. degree (Eng.) in electronics engineering from the Oaxaca Institute of Technology, Oaxaca, Mexico, and the M.Sc. and Ph.D. degrees in electronics and telecommunications from the CICESE Research Center, Ensenada, Mexico, in 2008 and 2016, respectively. He is currently a CONACyT Researcher assigned to CICESE Monterrey. His main research interests include wireless personal area networks, wireless sensor networks, low-power wide area networks, the Internet of Medical Things, e-health systems, radio over fiber architectures, embedded systems, and so on.

DIANA CARRIZALES-ESPINOZA received the B.Sc. degree (Eng.) in IT engineering from the Polytechnic University of Altamira, Mexico, in 2016. She is currently pursuing the master's degree with the Cinvestav Tamaulipas, Mexico. Her research interests include distributed systems and cloud storage.

JESUS CARRETERO (Senior Member, IEEE) is currently a Senior Full Time Professor-Researcher with the Computer Science and Engineering Department, Universidad Carlos III de Madrid, Madrid. He is also a Leader of the Research Group ARCOS, Universidad Carlos III de Madrid, and the Director of the Master in Administration and Management of information systems. He has published 17 books and 52 research projects. He has written 200 journal and congress articles. His major research lines are high-performance computing, cloud computing, parallel and distributed systems, computational linguistics, and real-time systems.