Internet of Things 5 (2019) 130–139

Contents lists available at ScienceDirect

Internet of Things

journal homepage: www.elsevier.com/locate/iot

Research article

Osmotic computing as a distributed multi-agent system: The Body Area Network scenario

Lorenzo Carnevale[a,b,*], Antonio Celesti[a,b], Antonino Galletta[a,b], Schahram Dustdar[c], Massimo Villari[a,b]

[a] MIFT Department, University of Messina, Via F. Stagno D'Alcontres 31, S. Agata, Messina 98166, Italy
[b] IRCCS Centro Neurolesi "Bonino-Pulejo", S.S.113 C/da Casazza, Messina, Italy
[c] Distributed Systems Group TU Wien, Argentinierstrasse 8, A-1040, Vienna, Austria

ARTICLE INFO

Article history:
Received 29 September 2018
Revised 4 January 2019
Accepted 6 January 2019
Available online 8 January 2019

Keywords:
Osmotic computing
Edge computing
Cloud computing
IoT
Orchestration
Elasticity
Artificial intelligence

ABSTRACT

Nowadays, the latest technological advancements have changed the centralized Cloud Computing model, going through Edge and Internet of Things (IoT), which are closer to end users. In particular, current Cloud Computing programming models deal with the recent evolution of the IoT phenomenon because smart devices are becoming more and more pervasive, powerful and inexpensive. Therefore, services need to be placed near such devices and resource orchestration techniques need to be redesigned. In this regard, Osmotic Computing aims at providing a new computing paradigm based on the deployment and migration strategies related to the infrastructures and applications requirements across Cloud, Edge, and IoT layers. The objective of this scientific work is to propose an Osmotic Computing architecture, based on a multi-agent system, according to a new software abstraction called MicroELement (MEL), that encapsulates resources, services and data necessary to run IoT applications. A Body Area Network (BAN) scenario is proposed in order to explore the Osmotic Computing potentiality and explain the reasons behind this new paradigm.

1. Introduction

In the last ten years, we observed an unstoppable growth of complementary technologies, such as Cloud Computing, Big Data, and Internet of Things (IoT). The latter has increased the connected devices number up to an estimate of 36 billion in 2021,[1] thanks to the miniaturization and low cost of these devices. Among these, almost a billion will be the wearable,[2] whereas about 3 billion will be the smartphone users.[3] Therefore, the amount of global IP data traffic in the same year is estimated at 280,000 petabytes per month[4] with the inevitable need to handle huge data amount from both migratory and

* Corresponding author at: MIFT Department, University of Messina, Via F. Stagno D'Alcontres 31, S. Agata, Messina 98166, Italy.
E-mail addresses: lcarnevale@unime.it, lcarnevale@irccsme.it (L. Carnevale), acelesti@unime.it (A. Celesti), angalletta@unime.it, antonino.galletta@irccsme.it (A. Galletta), dustdar@infosys.tuwien.ac.at (S. Dustdar), mvillari@unime.it, mvillari@irccsme.it (M. Villari).

[1] https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/.
[2] https://www.statista.com/statistics/487291/global-connected-wearable-devices/.
[3] https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/.
[4] https://www.statista.com/statistics/499431/global-ip-data-traffic-forecast/.

https://doi.org/10.1016/j.iot.2019.01.001
2542-6605/

computational point of view. Thus, the IoT model has become both a Big Data and a computation problem for each Cloud and Edge layer.

One of the sectors with the most growth and involvement in the Cloud, IoT, Big Data trinomial, is healthcare. Indeed, this industry is going further traditional storage and processing approaches, addressing new technological trends, such as Data Analytics [1]. A study of the McKinsey Global Institute [2] highlighted the Big Data penetration for healthcare, focusing on the potential to achieve insights.

Specifically, considering the Body Area Network (BAN) phenomenon, in which tens of sensors generate data streams concerning a patient not only for real time analysis of the health status, but also to fill electronic medical records. Indeed, the BAN is one of the best solution for matching telemedicine needs. Patients could be monitored in their familiar environment, limiting the home-hospital movement and the hospital bed reservation; involving doctors in an offline diagnosis system [3–5]. BAN involves the three technologies above mentioned. However, considering a complex environment in which the amount of patients' data streams grows up, receiving a real time alarm, about the health status, becomes a really Big Data challenge.

On the other hand, the 2017 saw these three technologies converge towards a single point of contact, called Osmotic Computing. It aims to identify, design, and implement a paradigm for managing data, resources and processes' software across IoT, Edge and Cloud systems, satisfying the end users' Quality of Service (QoS). The Cloud-Edge Computing integration [6] benefits have also been acknowledged by academic and industry initiatives, such as Cisco, Amazon AWS and the Open Fog Consortium.

Purpose of this scientific work is to present the progresses regarding the Osmotic Computing architecture, considering what discussed in our previous work [7]. Indeed, focusing on the microservices deployment aspect, the design of a distributed Osmotic orchestrator was investigated. In order to do that, we took advantage from the BAN issues, addressing its problems through our proposal. Therefore, the core of the orchestration process will be an Artificial Intelligence (AI) module that will learn through the monitoring of the Osmotic resources deployed on Cloud, Edge and/or IoT. Specifically, we investigated a multi-agent based approach for orchestrating microservices from the Cloud to Edge and IoT environments.

With this work, we aim to further move the discussion around the Osmotic Computing to the next level. Here, we lay the foundations to develop the heart of the topic, its orchestrator.

The rest of the paper is organized as follows. The section contains the related works. Section 3 presents the materials useful for dealing with the discussion about the Osmotic architecture: the BAN scenario and its problems are presented in Section 3.1, the nomenclature used in this paper is in Section 3.2 and an overview about Osmotic Computing is reported in Section 3.3. Section 4 is the discussion about the architecture; whereas conclusion are reported in Section 5

2. Related work

Osmotic Computing was introduced in 2016 as a new promising paradigm for the integration between a centralized Cloud layer and Edge, IoT layers [8], whereas its basic principles and enabling technologies were presented in [9]. Such a new paradigm, it could be used in different application scenarios requiring an intensive interaction between centralized Cloud systems and Edge devices. In [10] it was considered to design a Hospital Information System (HIS) interconnecting medical devices and patients' personal body networks with Hospital Cloud systems; other recent application fields regarded the efficient trust management in pervasive online social networks [11] and the management of IoT workflows [12].

One of the major issue of Osmotic Computing is the service orchestration management considering hybrid Cloud, Edge, and IoT systems. Up to now, service orchestration has been indipendently discussed considering both Cloud Computing [13–15], Edge Computing [16–18] and IoT [19]. For example, architecture for the dynamic management of end-to-end connections in a Cloud environment considering Software Defined Networking (SDN) technologies were investigated in [17,20–23]. An end-to-end SDN/NFV orchestration for video analytic using Edge and Cloud Computing over programmable optical networks is described in [24,25]. A piece of framework and novel computing models for Grid and Cloud service orchestration aimed at supporting scientists and business analysts at large scale were discussed in [26]. Dynamic resource orchestration for multi-task applications in heterogeneous mobile Cloud Computing are discussed in [27]. Here, the resource orchestration was formulated as multi-objective optimization problem considering energy consumption, cost, and availability metrics. A distributed framework for Cloud Computing orchestration able to manage the migration of Virtual Machines (VMs) was discussed in [28]. Platforms able to control high performance scientific workflow by means of Cloud applications were discussed in [29–31]. An orchestration engine based on a temporal reconfiguration approach that partitions the amount resources of Cloud servers proportionally between BPEL processes, applying a temporal partitioning algorithm, was discussed [32]. A SLA (Service Level Agreement) driven orchestration based methodology for Cloud Computing services was presented in [33]. The problem of security and privacy governance in Cloud Computing via SLAs and a policy orchestration service were investigated in [34]. The state of the art of container-based orchestration in Cloud, even considering future challenges, was discussed in [35].

Regarding service orchestration in IoT, the distributed orchestration in large-scale systems was discussed in [36]; whereas the requirements of a semantic based service orchestration were investigated in [37]. Intent-based management and orchestration of heterogeneous openflow/IoT SDN domains was discussed in [38], whereas testbed set-up for SDN orchestration across network Cloud and IoT domains was presented in [39]. The opportunities of applying an orchestration model in cognitive IoT solutions interconnecting instrumented worlds were discussed in [40]. An object-oriented model for object

orchestration in smart environments able to extend and customize smartphones was described in [41]. A model for trustworthy orchestration in IoT using a public/subscribe approach and MQTT was presented in [42]. A scalable piece of framework for the provisioning large-scale IoT deployments was discussed in [43]. The orchestration in distributed web-of-objects for creation of user-centered IoT services was discussed in [44–46].

Service orchestration in Edge Computing is a rather new topic. The deployment orchestration of microservices with geographical constraints using OpenStack Heat components was discussed in [47]; whereas an end-to-end SDN/NFV orchestration for video analytic using Edge and Cloud Computing over programmable optical networks was discussed in [24]. An adaptive orchestration platform called ECHO for hybrid dataflows across Cloud and Edge was described in [48]. In particular, the ECHO's hybrid dataflow composition is able to operate on diverse data models such as streams, micro-batches and files, and interface with native run-time engines like TensorFlow and Storm to execute them. An architecture able to move IoT applications on Edge Computing layers was described in [49]. A SDN/NFV orchestration of 5G services in hybrid Cloud multi-domain networks was discussed in [50]. A preliminary discussion about the use of container virtualization to orchestrate Edge Computing environments was discussed in [51]. A method to enable ubiquitous Mobile Edge Computing (MEC) has been considered in [52]; Mobile Edge servers are located at the edge of the mobile network, removing the need to use Cloud Computing for intensive computation and storage tasks.

On the other hand, focusing on the use case developed in this scientific work, the BAN necessity for looking through Cloud platforms are several. The first challenge is accessing the data, capturing the sensors' streams; therefore, data integration is required to address the variety of structures and formats; instead, for a reliable analysis, a cleaned dataset is really important; as well the security rules; finally, data can be analyzed and organized in user friendly views. Any aspect above mentioned falls in the Big Data domain and therefore, working with BAN means defining the typical Data Analytics workflow. Indeed, each of these is part of the Big Data workflow, as explained in [53]. Therefore, the execution does not matter where is performed, but only that the output of a step is the input of the next one.

Clearly, solutions have been presented about the idea to improve the BAN quality using Cloud platforms for managing streams. Among these, Fortino et al. compared in [54] the ECGaaS, Cloud BAN e-Health and BodyCloud platforms. The first one [55] provides an ECG Data Analytics service, in which patients' records are collected in order to perform a real time ECG beat analysis. The Cloud Ban e-Health [56] supports data streams collection from hospitalized and at home patients in order to monitor the health status. Finally, BodyCloud [57,58] is a SaaS solution which support management processing and analysis of body sensor data streams. The interesting innovation regards a workflow-oriented decision support to take actions according to the analyzed data.

Differently from the aforementioned scientific works, in this paper we focus on the Osmotic Computing solution for addressing the Cloud, Edge and IoT integration issue. The idea is to propose a possible solution to the BAN problem regarding the integration with Cloud Computing. Specifically, we have investigated a multi-agent based approach for orchestrating microservices from the Cloud to Edge and IoT environments. Therefore, we aimed to improve the literature about that topic, moving the discussion even closer to the practical vision. Thus, in the next pages a discussion about how to design an Osmotic orchestrator is faced.

Regarding security, it is absolutely a main topic considering Cloud, Edge and IoT communication, mostly for our solution. However, this subject is not focus of this scientific work. For sure, before to start the implementation of the proposed architecture, it will be extended with all the security needed knowledge. On our side, we already started the study of a solution to securely manage hybrid Cloud-Edge environments in [6], based on an instant-message communication solution; further, we defined the Software Defined Membrane [59] as the main component responsible to orchestrate the osmotic transfer of MicroELements, basing on Private Blockchain technologies. On the other hand, the scientific community is rich of valuable works. In [60], a survey of IoT and Cloud Computing security issues has been presented. Instead, a location privacy-preserving algorithm with good advantages in term of lower probability of re-leaving the user's location has been presented in [61].

3. Material

3.1. Scenario

This Section explains why the Osmotic Computing could become an important research point for the next years, going through the BAN scenario issues. Referring to Fig. 1, our scenario involves several Body Sensor Units (BSU), such as IoT devices, applied on the patients' body; in addition, a single Body Control Unit (BCU) gathers data produced by the devices, with the purpose of forwarding to the remote nodes. Indeed, about the latter, a distributed hospital system has the technologies to collect and analyze these data; specifically, this scenario considers any building as a Edge node. Our research background about the matter comes from the experience matured at the IRCCS Centro Neurolesi "Bonino Pulejo" (Messina, Italy). It is a Scientific Institute for Recovery and Care with the mission in the field of neuroscience for the prevention, recovery and treatment of individuals with severe acquired brain injury, besides spinal cord and neurodegenerative diseases, by integrating highly specialized healthcare, technological innovation and higher education. The IRCCS Institute is spread among a couple of buildings and, as shown in Fig. 1, these are the remote computing nodes involved in our scenario.

As said in Section 1, this scenario requires the definition of an Analytic workflow [62]. Referring to a real use case encountered during our research at the IRCCS Institute, the patients have been dressed with a set of BSU such as

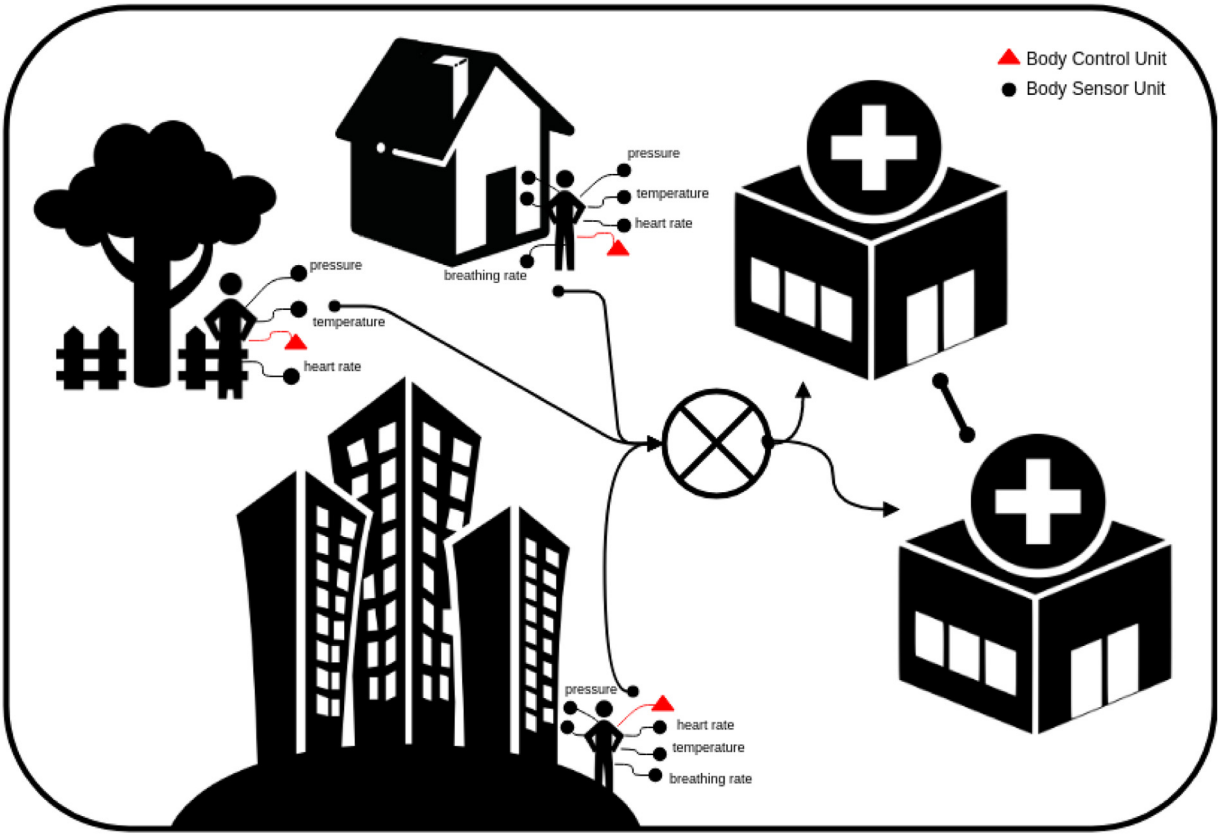

Fig. 1. Body Area Network traditional scenario.

electrocardiography (ECG), blood pressure (BP), body temperature (BT), heart rate (HR) and breathing rate (RR). Data produced by those are collected by a BCU, which is a Microprocess unit-like device, and forwarded to the hospital Edge Computing system. Thus, data need to be integrated going over the different formats. Perhaps, sensors generate XML and JSON data, that are parsed according with the database structure in which should be stored. For helping the data scientists' work, parsed data are cleaned by inconsistent entries; as well cleaned by not meaningful values. Therefore, data are organized in the database in order to be ready for a comparison with those coming from other sources. Finally, analysis and visualization phases are useful for triggering the real time alarm about the patient's health status.

What is really important, regarding the described scenario, is the microservice nature of each above mentioned step. This means that the phases are independent from platform and network, small in size development and short in time deployment. Moreover, the questions are: what about a huge scenario with many BANs? What if an intensification of the reading raises a health status alarm? It should increase in terms of patients and data streams and, therefore, overload the communications from/to the Edge nodes, causing a not well timed behavior.

3.2. Osmotic MELs

The convergence between Cloud Computing, Edge, and IoT requires an Osmotic management of resources, services, and data, whose elements can move across different heterogeneous infrastructures. More specifically, referring to Fig. 2, IoT applications deployed in distributed environments can be viewed as a graph of MicroELementS (MELs), composed of:

- MicroServices (MS) for implementing specific functionalities, which can be deployed and migrated across the virtualized infrastructures;
- MicroData (MD) for representing information flows from/to IoT devices, which can have in different data formats.

The MELs graph needs to be orchestrated across Cloud, Edge, and IoT according to specific QoS requirements. Let us remark that MELs are not physical resources, but represent software and data abstractions. According to Fig. 2, the leaf node is represented by MicroUserService (MUS, i.e., an IoT application) and MicroOperationalService (MOS, i.e., an Operating System) along with MicroUserData (MUD, i.e., User Data) and MicroOperationalData (MOD, i.e., MS configuration). The MELs are smartly deployed on Cloud and Edge in virtual components, such as lightweight containers (e.g., Docker, Google Container, Amazon Compute Cloud Container, etc.); whereas uPython-VM, uLUA-VM, Javascript on IoT have emerged as a lightweight

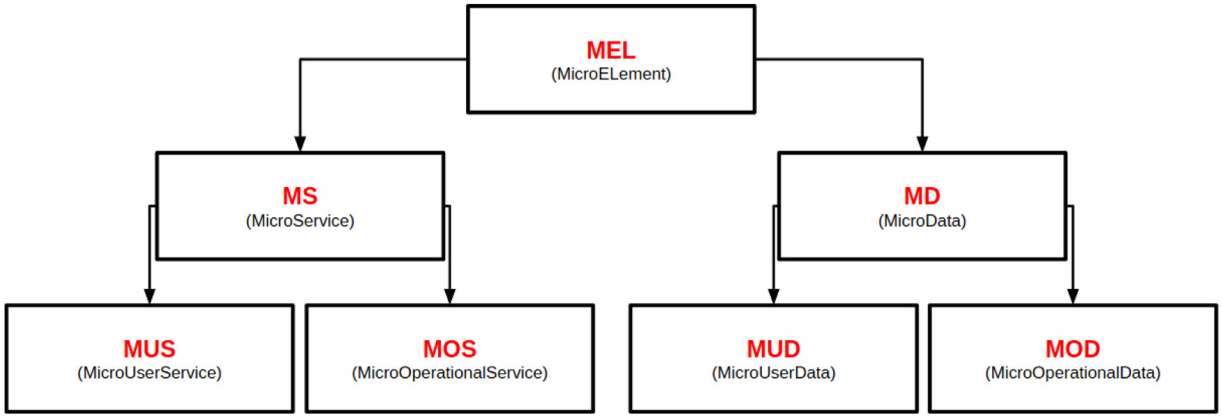

Fig. 2. MicroElementS hierarchy.

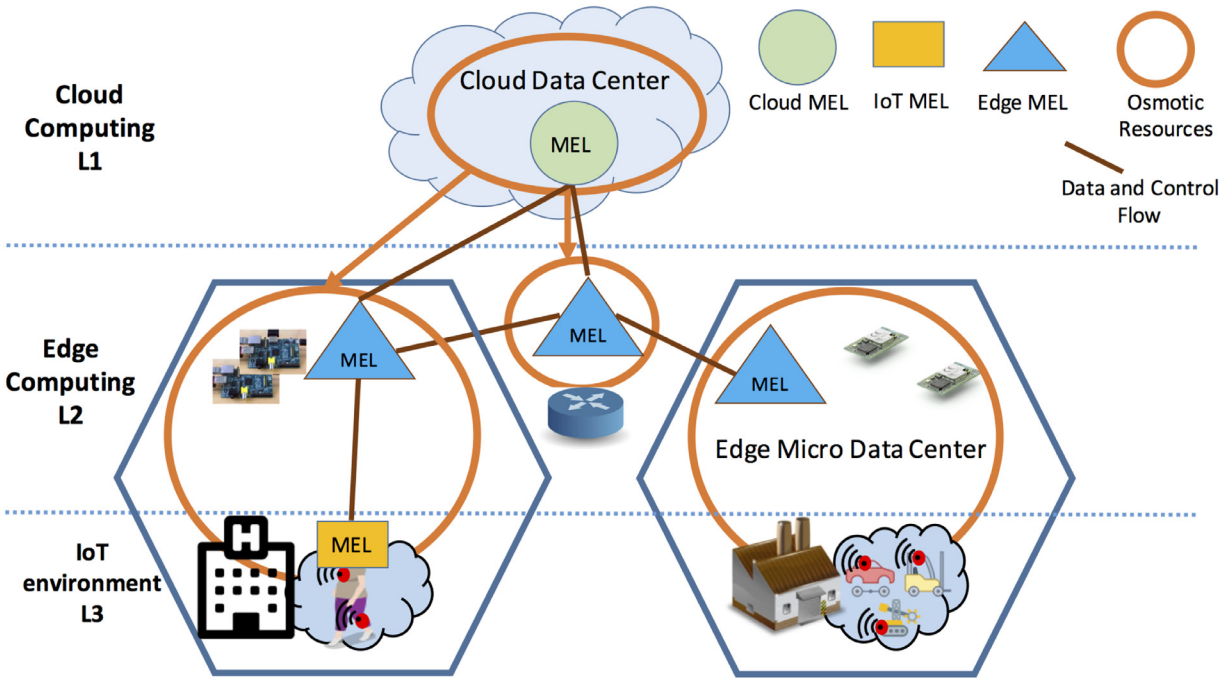

Fig. 3. Osmotic computing scenario.

alternative to the hypervisor-based approach used in the Cloud Data Center. A container encapsulates only well-defined software components (e.g., database servers), reducing the deployment overhead and increasing instance density on a single device. For instance, an example of MS could be a Docker Container and a MD contains JSON-based metadata. Moreover, MD can be both passive data (can be read or updated on devices) and active (can be queried), e.g., MD stored in NoSQL DBs as MongoDB, Cassandra, etc.

3.3. Osmotic computing platform

Borrowing the term from chemistry, "osmosis" represents the MELs spread across the Cloud Data Center (CDC) and the Edge micro Data Center (EmDC). The Osmotic Computing overcomes the MELs elastic management concept, since the deployment and migration strategies are related to the requirements of both infrastructures (i.e., load balancing, reliability, availability) and applications (i.e., detection, implementation, awareness of the context, proximity, QoS). Moreover, in order to overcome the heterogeneity of IoT resources, the MEL abstraction allows us to support a virtual environment that can be adapted according to the available hardware equipment. Looking at Fig. 3, the Layer 3 (L3), the one closest to end users and/or physical entities, shows how MELs are deployed in embedded devices. L3 IoT devices communicate according to standardized protocols, such as CoAP (Constrained Application Protocol), supported by the RESTful interface.

Furthermore, Fig. 3 shows MELs (at L2) deployed in different embedded devices (i.e., IoT Gateway such as Raspberry Pi 3). Gateway nodes perform operations (mean, min, max, filtering, aggregation, etc.) on the data flow acquired from L3. More often, these devices acquire data with a predefined frequency, depending on specific system requirements and the device ability to gather data.

In Fig. 3, a more complex computational and storage capability is available to MELs at L1, allowing to perform simulations and/or analyses on data.

The Cloud, Edge and IoT infrastructure also has its own target function which influences the performed operations. For example, Edge (L2 in Fig. 3) generally includes devices with limited resources (i.e., limited battery power, network range, etc.) which must perform operations by these constraints. Therefore, the storage and compute capacity in the Edge must be shared among multiple concurrent data streams (probably from L3 in Fig. 3), limiting the analysis to the streams number and time constraints. Cloud operations (L1 in Fig. 3) are based on pre-agreed goals between a client and a data center provider, such as throughput, response times, costs, etc. It is a key research challenge for real-time streaming applications understanding how an application hosted on a Cloud in L1 can interact and coordinate with the IoT (in L3) and the Edge (in L2). Driven by QoS requirements, the MELs can be distributed among Cloud, Edge and IoT. The distribution of data analysis through these different infrastructures can improve the overall performance of the IoT application and reduce the load on the main network.

Orchestration in the Osmotic environment smartly configures the movement and deployment of MELs in response to QoS, security/privacy requirements and runtime requests. Indeed, static and fixed approaches are not able to provide IoT solutions. The Osmotic Computing aims to abstract the services (MELs) and the infrastructure (IoT, Cloud, Edge) to decouple the applications from the hardware and enable the possibility to flow the services from Cloud to Edge and/or IoT and viceversa.

For example, considering a situation in which there are many IoT devices, which in particular situations are collecting large volumes of data on L3. Furthermore, considering that given the stability and capacity of the network, the amount of data produced and their subsequent transmission to a Cloud (L1) are unsustainable from the network point of view. If these data were to be analyzed in the Cloud, this would be unrealistic and the current system may not be able to continue. Using the Osmotic approach, when a bottleneck of this type is detected, a Smart Orchestrator moves the processing of some data to the Edge (L2).

4. Osmotic agents

The Sections already developed provide a background about the Osmotic Computing and the reasons behind the introduction of that.

Our aim is to use the Osmotic Computing for resolving the problem highlighted in the BAN scenario. First of all, we would like to redesign the scenario implementing the Osmotic concepts; therefore, we discuss how to design an architecture that enables what has been wrote so far.

The question we want to answer now is: may Osmotic Computing improves the BAN scenario, explained in Section 3.1? In order to give an answer, let us assume the following points:

- the Hospital System is an Edge layer composed by several physical machines. It means the IRCCS Institute's buildings are federated for creating an Edge layer. It represents the big computation node of the Osmotic architecture; and is responsible for the support execution;
- the BCU is another Edge layer composed by one physical machine. It is a lightweight computation node that manages the BAN, allowing the in-local execution of the BSUs' generated data;
- each physical machine is an agent. Therefore, the architecture is a distributed multi-agent system spread in two Edge layers, just considering the point-to-point ideal case. On the other hand, in the complex case, in which more BANs communicates with the Hospital System, the communication is only between each BSU agent and the Hospital System agents. For privacy reasons, the communication among BCUs is not allowed.

Thus, from a general point of view, we assume the workflow starts inside the BCU agents. Consequently, the BCU Edge layers could perform each operation required by the scenario, without the Hospital Edge layer support. However, if a step is overloaded, than the BCU moves that to the Hospital System agents. In this context, the microservices choice is clear because of the lightweight migration nature of these.

4.1. Characteristics

The architecture has been proposed in this scientific work is a distributed multi-agent system. It is a set of agents distributed among the Edge and Cloud layers, that is the environment, interacting through a specific arrangement (i.e., BCU agents to Hospital agents). The agent independently works in the environment, monitoring the latter about the microservices overloading. In this context, a prompt awareness of the necessity to migrate a microservice is provided by means of artificial intelligence algorithms for relocating the process, perhaps through a microservice redeployment.

Each agent is self-orchestrated, which means the agents are able to manage the workflow as a set of microservices. The agent could use containers or vms manager for deploying the microservices. Moreover, it monitors each agent about the

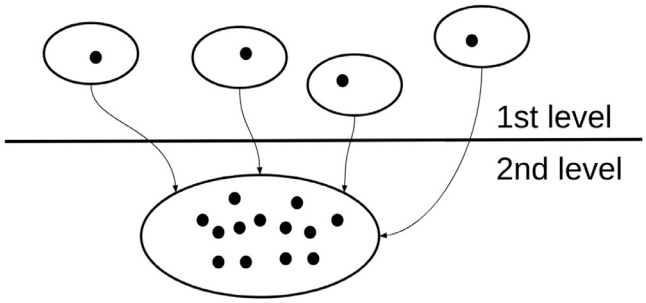

Fig. 4. Osmotic structure referred to the BAN scenario.

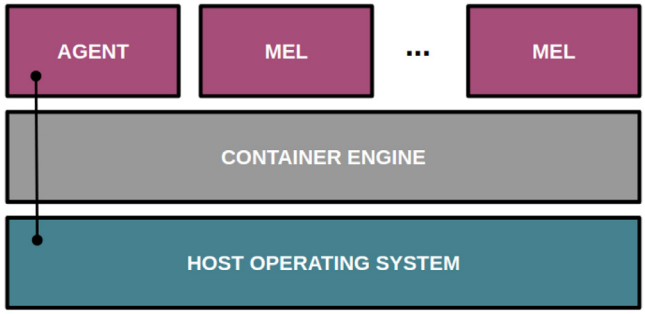

Fig. 5. High level design of the Osmotic Agent.

overloading for allowing the microservice migration to another agent. In this context, an opened question regards which is the owner of the migrated microservice. Is it the agent in which the microservice will migrate or the agent which ask for that? What happened when a microservice needs to migrate again to a third agent? In order to give an explanation and simplify this problem, let us now considering the logical structure in Fig. 4.

Up the line, there are the 1st level agents, designed as the nearest ones to the data sources. Considering the scenario in Section 3.1, these are the BCUs; that are set one by one in Fig. 4. Under the line, there are the 2nd level agents, which are the remote nodes with high performance, designed for supporting the 1st level agents. In our scenario, these are the Hospital System agents, collected together in a federated environment. In this new context, we aim to assign the management of the workflow to the 1st level agents, in which the microservices has been deployed the first time. In this way, these agents could track the microservices life cycle and maintain their ownership.

Considering the microservices as lightweight processes embedded in containers, the flow management over the network is handled through an overlay network driver. Docker is one of the service provider; and it is able to automatically route packets to/from hosts.

4.2. Agent orbit

What has been discussed so far clarifies the hierarchy of the agents distributed among the layers; and provides key points in the evolution design. However, the most important question, regarding the migration trigger, is still opened. Considering the scenario discussed in this paper, the prompt response in the health status alarm generation is the main Key Performance Indicator (KPI). Translating it in functionalities our agents could perform, the migration exists according to the microservice's response time. Moreover, considering a Service Level Agreement (SLA), if the analyzed (i.e., predicted) response time affects the SLA, then the microservices has to be migrated.

Thus, the response time is the barycenter of the agent design, which we propose driven by a Monitor, Analyze, Plan and Execute (MAPE) loop. Agent monitors the microservices; it analyses (i.e, predict next) the microservices' monitored data; 1st level agent plans to migrate or not a microservice to another 2nd level agent; 2nd level agent plans to communicate to 1st level agent the microservice overloaded; finally, agent executes the plan.

4.3. Agent design

According to the characteristics discussed so far, the first design tentative is showed in Fig. 5. This represents the first ever designed Osmotic Agent, which task is to mark devices (i.e. microprocessors, vms or physical machines) through the installation of a software that enables communication from/to other agents. From a practical point of view, this component

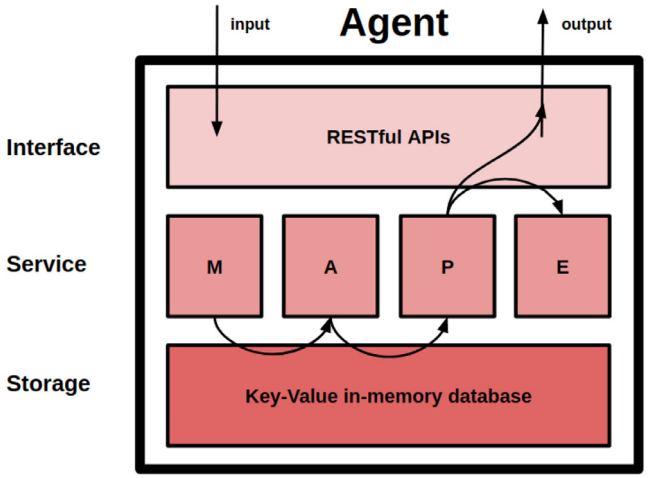

Fig. 6. Logical architecture of the Osmotic Agent.

Table 1
How the Executor runs microservices and microservices workflow.

Run microservice	Run microservices workflow
It receives and downloads the MS Repository URL	
It receives and downloads the MUD	
It composes and runs the MEL	
	It composes and runs the workflow

should be a lightweight vm, that interacts with the host operating system in order to enable the MAPE loop; i.e., it monitors itself and other vm and deploys MELs on the same level.

Referring to Fig. 6, the Osmotic Agent is designed on three layers. The Interface layer includes the RESTful APIs for using the MAPE functionalities. Specifically, it is a HTTP interface useful for enabling the MELs creation and migration; and a bidirectional communication among the agents. The Storage layer includes an In-Memory database that implements the Publish/Subscribe messaging paradigm. It works as message broker, in which the MAPE components, in the Service layer, are publisher and/or subscriber. Finally, the Service layer includes the MAPE loop features, that are explained in detail in the following.

The Execute component is the starting point of our orbit. Considering the workflow discussed in the scenario, it has two ways: the Executor could run independent microservices, which not affect others if migrated; otherwise, it could run microservices' workflow, in which the migration generates a reconfiguration of the microservice located one step before. In order to enable that, let us assume:

- MS as MUS + MOS, a container with application;
- MUD as a JSON configuration file;
- MEL as MS+ MUD;
- workflow as MELs composition.

The operations to complete the execution of both ways are summarized in Table 1, in which the microservice only running is a special case of the microservices' workflow. Indeed, firstly, the Executor receives the MS Repository URL and downloads the MS from there. The idea is to limit the network load during the migration, moving inside a message just an URL, that is a string; therefore, the Executor can independently downloads the right container version. In the same way, it receives and downloads the MUD; and, next, the Executor composes the MEL as a container with its configuration file. Finally, considering the workflow requirements, the Executor composes the MELs in a pipeline before to run it.

The Monitor component has an important task in the MAPE loop and mostly in the dynamic migration. How already discussed in Section 4.2, we aim to use the response time as KPI of our loop, considering the scenario. This component has to monitor the response time of each microservice, in order to study the past and present values, according with the environment. The hard part of the response time monitoring is the necessity to query the microservices by means of a tool not already present in those. Our idea is to inject a serverless-like application. More specifically, for each microservice, we aim to inject a lightweight one-shot web service, which performs a HTTP GET. Therefore, the response time of that will be calculated and collected. Thanks to it the Monitor component collects the response times, which values are mandatory to build a dataset useful to the Analysis component.

For the Analysis component, we propose two Machine Learning techniques that could cover our analytic need. A deeper discussion about it will be deal with the future works, but now let us talk just about the concept. One of the techniques is the Reinforcement Learning. It is a sub area of Machine Learning, which aims to learn through the environment's observations, in order to adapt itself by means of a long term reward. The latter is just a performance evaluation, which, considering our scenario, regards the response time evaluation in the future. Thus, according to SLA, it generates or not the migration. The most common Reinforcement Learning algorithm is the Q-learning, but Deep Reinforcement Learning techniques have been developed in the last years. On the other hand, a Time Series Analysis could be another solution for our purpose. It is a discrete time sequence used to create model able to predict future value based on the observed values. Also in this case, considering our scenario, this method could help us to predict the future value of the response time that, according to SLA, generates or not the migration.

Finally, the Plan component uses the analysis results for performing two tasks:

- nothing if the migration is not necessary;
- asks for 2nd level agents available for the microservice hosting and then invokes the Execute component for enabling the migration.

5. Conclusion

This paper investigated how to design an Osmotic Computing architecture through a distributed multi-agent system. We demonstrate a scenario such as the Body Area Network (BAN) to prove the potentiality of the new computational paradigm. We focused primarily on the Osmotic Computing concepts in order to provide the base of the paradigm; then a new Osmotic Computing BAN scenario has been proposed for describing the components of the architecture. Specifically, a MAPE loop is how we have designed a multi-agent able to enable the new paradigm.

As future work, we aim at validating our architecture through an implementation and a comparison with the existing others.

Acknowledgments

This work has been supported by Cloud for Europe (C4E) Tender: *REALIZATION OF A RESEARCH AND DEVELOPMENT PROJECT (PRE-COMMERCIAL PROCUREMENT) ON "CLOUD FOR EUROPE"*, Italy-Rome: Research and development services and related consultancy services Contract notice: 2014/S 241-424518. Directive: 2004/18/EC. (http://www.cloudforeurope.eu/).

References

[1] L. Carnevale, R.S. Calabr, A. Celesti, A. Leo, M. Fazio, P. Bramanti, M. Villari, Towards improving robotic-assisted gait training: can big data analysis help us? IEEE Internet Things J. (2018) 1, doi:10.1109/JIOT.2018.2855937.
[2] McKinsey Global Institute, Big Data: The Next Frontier for Innovation, Competition, and Productivity, June 2011. McKinsey Global Institute.
[3] A. Celesti, M. Fazio, A. Romano, A. Bramanti, P. Bramanti, M. Villari, An oais-based hospital information system on the cloud: analysis of a NoSQL column-oriented approach, IEEE J. Biomed. Health Inf. 22 (3) (2018) 912–918, doi:10.1109/JBHI.2017.2681126.
[4] D. Mulfari, A. Celesti, A. Puliafito, M. Villari, How cloud computing can support on-demand assistive services, in: Proceedings of the 10th International Cross-Disciplinary Conference on Web Accessibility, in: W4A '13, ACM, New York, NY, USA, 2013, pp. 27:1–27:4, doi:10.1145/2461121.2461140.
[5] M. Fazio, A. Celesti, F.G. Mrquez, A. Glikson, M. Villari, Exploiting the FIWARE cloud platform to develop a remote patient monitoring system, in: Proceedings of the IEEE Symposium on Computers and Communication (ISCC), 2015, pp. 264–270, doi:10.1109/ISCC.2015.7405526.
[6] A. Celesti, M. Fazio, A. Galletta, L. Carnevale, J. Wan, M. Villari, An approach for the secure management of hybrid CloudEdge environments, Fut. Gen. Comput. Syst. 90 (2019) 1–19, doi:10.1016/j.future.2018.06.043.
[7] L. Carnevale, A. Celesti, A. Galletta, S. Dustdar, M. Villari, From the cloud to edge and IoT: a smart orchestration architecture for enabling osmotic computing, in: Proceedings of the 32nd International Conference on Advanced Information Networking and Applications Workshops (WAINA), IEEE, 2018, doi:10.1109/waina.2018.00122.
[8] M. Villari, M. Fazio, S. Dustdar, O. Rana, R. Ranjan, Osmotic computing: a new paradigm for edge/cloud integration, IEEE Cloud Comput. 3 (6) (2016) 76–83.
[9] M. Villari, A. Celesti, M. Fazio, Towards osmotic computing: looking at basic principles and technologies, Adv. Intell. Syst. Comput. 611 (2018) 906–915.
[10] L. Carnevale, A. Galletta, A. Celesti, M. Fazio, M. Paone, P. Bramanti, M. Villari, in: Big data his of the IRCCS-ME future: the osmotic computing infrastructure, 189, 2018, pp. 199–207.
[11] V. Sharma, I. You, R. Kumar, P. Kim, Computational offloading for efficient trust management in pervasive online social networks using osmotic computing, IEEE Access 5 (2017) 5084–5103.
[12] M. Nardelli, S. Nastic, S. Dustdar, M. Villari, R. Ranjan, in: Osmotic flow: osmotic computing + IoT workflow, 4, 2017, pp. 68–75.
[13] F. Mrmol, M. Kuhnen, Reputation-based web service orchestration in cloud computing: a survey, Concurr. Comput. 27 (9) (2015) 2390–2412.
[14] R. Ranjan, R. Buyya, S. Nepal, D. Georgakopulos, A note on resource orchestration for cloud computing, Concurr. Comput. 27 (9) (2015) 2370–2372.
[15] M. Katyal, A. Mishra, in: Orchestration of cloud computing virtual resources, 2014, pp. 833–838.
[16] Y. Jiang, Z. Huang, D. Tsang, Challenges and solutions in fog computing orchestration, IEEE Netw. (2017).
[17] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, D. Sabella, On multi-access edge computing: a survey of the emerging 5g network edge cloud architecture and orchestration, IEEE Commun. Surv. Tutor. 19 (3) (2017) 1657–1681.
[18] Z. Wen, R. Yang, P. Garraghan, T. Lin, J. Xu, M. Rovatsos, Fog orchestration for internet of things services, IEEE Internet Comput. 21 (2) (2017) 16–24.
[19] C. Consel, M. Kabac, in: Internet of things: from small-to large-scale orchestration, 2017, pp. 1748–1755.
[20] A. Mayoral, R. Vilalta, R. Muoz, R. Casellas, R. Martnez, SDN orchestration architectures and their integration with cloud computing applications, Opt. Switch. Netw. 26 (2017) 2–13.
[21] R. Bonafiglia, G. Castellano, I. Cerrato, F. Risso, in: End-to-end service orchestration across SDN and cloud computing domains, 2017.
[22] Y. Kim, S. Kang, C. Cho, S. Pahk, in: SDN-based orchestration for interworking cloud and transport networks, 2016, pp. 303–307.
[23] A. Mayoral, R. Vilalta, R. Muoz, R. Casellas, R. Martnez, in: Performance analysis of SDN orchestration in the cloud computing platform and transport network of the adrenaline testbed, 2015. 2015-August

[24] R. Vilalta, I. Popescu, A. Mayoral, X. Cao, R. Casellas, N. Yoshikane, R. Martnez, T. Tsuritani, I. Morita, R. Muoz, in: End-to-end SDN/NFV orchestration of video analytics using edge and cloud computing over programmable optical networks, 2017.
[25] A. Galletta, A. Cuzzocrea, A. Celesti, M. Fazio, M. Villari, A scalable cloud-edge computing framework for supporting device-adaptive big media provisioning, in: Proceedings of the 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID), 2018, pp. 669–674, doi:10.1109/CCGRID.2018.00099.
[26] F. Messina, R. Mikkilineni, G. Morana, Middleware, framework and novel computing models for grid and cloud service orchestration, Int. J. Grid Util. Comput. 8 (2) (2017) 71–73.
[27] Q. Qi, J. Liao, J. Wang, Q. Li, Y. Cao, in: Dynamic resource orchestration for multi-task application in heterogeneous mobile cloud computing, 2016, pp. 221–226. 2016-September
[28] R. Weingartner, G. Brascher, C. Westphall, in: A distributed autonomic management framework for cloud computing orchestration, 2016, pp. 9–17.
[29] J. de Carvalho Silva, F. de Carvalho Junior, A platform of scientific workflows for orchestration of parallel components in a cloud of high performance computing applications, Lect. Notes Comput. Sci. (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 9889 LNCS (2016) 156–170.
[30] L. Miroslaw, M. Pantic, H. Nordborg, in: Unified cloud orchestration framework for elastic high performance computing in the cloud, 2016, pp. 291–298.
[31] K. Shams, M. Powell, T. Crockett, J. Norris, R. Rossi, T. Soderstrom, in: Polyphony: a workflow orchestration framework for cloud computing, 2010, pp. 606–611.
[32] Z. Brahmi, C. Gharbi, Temporal reconfiguration-based orchestration engine in the cloud computing, Lect. Notes Bus. Inf. Process. 176 LNBIP (2014) 73–85.
[33] M. Irfan, Z. Hong, N. Aimaier, L. Zhuguo, SLA (service level agreement) driven orchestration based new methodology for cloud computing services, Adv. Mater. Res. 660 (2013) 196–200.
[34] M. Mont, K. McCorry, N. Papanikolaou, S. Pearson, in: Security and privacy governance in cloud computing via SLAs and a policy orchestration service, 2012, pp. 670–674.
[35] A. Tosatto, P. Ruiu, A. Attanasio, in: Container-based orchestration in cloud: state of the art and challenges, 2015, pp. 70–75.
[36] E. Yigitoglu, L. Liu, M. Looper, C. Pu, in: Distributed orchestration in large-scale IoT systems, 2017, pp. 58–65.
[37] E. Chindenga, M. Scott, C. Gurajena, in: Semantics based service orchestration in IoT, Part F130806, 2017.
[38] W. Cerroni, C. Buratti, S. Cerboni, G. Davoli, C. Contoli, F. Foresta, F. Callegati, R. Verdone, in: Intent-based management and orchestration of heterogeneous openflow/ioT SDN domains, 2017.
[39] S. Fichera, M. Gharbaoui, P. Castoldi, B. Martini, A. Manzalini, in: On experimenting 5g: testbed set-up for SDN orchestration across network cloud and IoT domains, 2017.
[40] C. Li, F. Darema, V. Chang, Distributed behavior model orchestration in cognitive internet of things solution, Enter. Inf. Syst. (2017) 1–21.
[41] L. Bergesio, A. Bernardos, J. Casar, in: An object-oriented model for object orchestration in smart environments, 109, 2017, pp. 440–447.
[42] M. Bottone, G. Primiero, F. Raimondi, V. De Florio, in: A model for trustworthy orchestration in the internet of things, 2016, pp. 171–174.
[43] M. Vögler, J.M. Schleicher, C. Inzinger, S. Dustdar, A scalable framework for provisioning large-scale IoT deployments, ACM Trans. Internet Technol. 16 (2) (2016) 11:1–11:20.
[44] N. Lee, H. Lee, W. Ryu, K. Heo, in: Web of object service architecture for device orchestration and composition, 2014.
[45] Y. Kim, S. Lee, Y. Jeon, I. Chong, S. Lee, in: Orchestration in distributed web-of-objects for creation of user-centered IoT service capability, 2013, pp. 750–755.
[46] Y. Kim, S. Lee, I. Chong, Orchestration in distributed web-of-objects for creation of user-centered IOT service capability, Wirel. Pers. Commun. 78 (4) (2014) 1965–1980.
[47] M. Villari, A. Celesti, G. Tricomi, A. Galletta, M. Fazio, in: Deployment orchestration of microservices with geographical constraints for edge computing, 2017, pp. 633–638.
[48] P. Ravindra, A. Khochare, S. Reddy, S. Sharma, P. Varshney, Y. Simmhan, Echo: an adaptive orchestration platform for hybrid dataflows across cloud and edge, Lect. Notes Comput. Sci. (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 10601 LNCS (2017) 395–410.
[49] A. Hegyi, H. Flinck, I. Ketyko, P. Kuure, C. Nemes, L. Pinter, in: Application orchestration in mobile edge cloud : Placing of IOT applications to the edge, 2016, pp. 230–235.
[50] R. Vilalta, A. Mayoral, R. Casellas, R. Martinez, R. Munoz, in: SDN/NFV orchestration of multi-technology and multi-domain networks in cloud/fog architectures for 5g services, 2016.
[51] S. Hoque, M. Brito, A. Willner, O. Keil, T. Magedanz, in: Towards container orchestration in fog computing infrastructures, 2, 2017, pp. 294–299.
[52] Y. Jararweh, M. Alsmirat, M. Al-Ayyoub, E. Benkhelifa, A. Darabseh, B. Gupta, A. Doulat, Software-defined system support for enabling ubiquitous mobile edge computing, Comput. J. 60 (10) (2017) 1443–1457, doi:10.1093/comjnl/bxx019.
[53] Informatica, The Big Big Data Workbook, Informatica.
[54] G. Fortino, G.D. Fatta, M. Pathan, A.V. Vasilakos, Cloud-assisted body area networks: state-of-the-art and future challenges, Wirel. Netw. 20 (7) (2014) 1925–1938, doi:10.1007/s11276-014-0714-1.
[55] S. Pandey, W. Voorsluys, S. Niu, A. Khandoker, R. Buyya, An autonomic cloud environment for hosting ECG data analysis services, Fut. Gen. Comput. Syst. 28 (1) (2012) 147–154, doi:10.1016/j.future.2011.04.022.
[56] A. Lounis, A. Hadjidj, A. Bouabdallah, Y. Challal, Secure and scalable cloud-based architecture for e-health wireless sensor networks, in: Proceedings of the International Conference on Computer Communication Networks (ICCCN), Munich, Germany, 2012, p. 1.
[57] G. Fortino, G.D. Fatta, S.F. Ochoa, C.E. Palau, Engineering future interoperable and open IoT systems, J. Netw. Comput. Appl. 81 (2017) 59–61, doi:10.1016/j.jnca.2017.01.017.
[58] G. Fortino, D. Parisi, V. Pirrone, G.D. Fatta, BodyCloud: a SaaS approach for community body sensor networks, Fut. Gen. Comput. Syst. 35 (2014) 62–79, doi:10.1016/j.future.2013.12.015.
[59] M. Villari, A. Galletta, A. Celesti, L. Carnevale, M. Fazio, Osmotic computing: software defined membranes meet private/federated blockchains, in: Proceedings of the IEEE Symposium on Computers and Communications (ISCC), IEEE, 2018, doi:10.1109/iscc.2018.8538546.
[60] C. Stergiou, K.E. Psannis, B.-G. Kim, B. Gupta, Secure integration of IoT and cloud computing, Fut. Gen. Comput. Syst. 78 (2018) 964–975, doi:10.1016/j.future.2016.11.031.
[61] G. Sun, V. Chang, M. Ramachandran, Z. Sun, G. Li, H. Yu, D. Liao, Efficient location privacy algorithm for internet of things (IoT) services and applications, J. Netw. Comput. Appl. 89 (2017) 3–13, doi:10.1016/j.jnca.2016.10.011.
[62] L. Carnevale, A. Galletta, A. Celesti, M. Fazio, M. Paone, P. Bramanti, M. Villari, Big data his of the IRCCS-me future: the osmotic computing infrastructure, in: A. Longo, M. Zappatore, M. Villari, O. Rana, D. Bruneo, R. Ranjan, M. Fazio, P. Massonet (Eds.), Cloud Infrastructures, Services, and IoT Systems for Smart Cities, Springer International Publishing, Cham, 2018, pp. 199–207.