Platforms for Smart Environments and Future Internet Design: A Survey

ANTONIO MARCOS ALBERTI[1], **MATEUS A. S. SANTOS**[2], **RICARDO SOUZA**[2],
HIRLEY DAYAN LOURENÇO DA SILVA[2], **JORGE ROBERTO CARNEIRO**[1],
VITOR ALEXANDRE CAMPOS FIGUEIREDO[1], **AND**
JOEL J. P. C. RODRIGUES[3,4] **(Senior Member, IEEE)**

[1]National Institute of Telecommunications, Santa Rita do Sapucaí 37540-000, Brazil
[2]Ericsson Research, Indaiatuba 13330-050, Brazil
[3]Department of Electrical Engineering, Federal University of Piauí, Teresina 64049-550, Brazil
[4]Instituto de Telecomunicações, 1049-001 Lisboa, Portugal

Corresponding author: Antonio Marcos Alberti (alberti@inatel.br)

ABSTRACT Internet of things (IoT) is pushing the integration of physical and virtual worlds. Sensor devices provide rich sensory information to context-aware services. Actuators implement in the physical world application needs. The role of software is increasing as more and more resources become virtualized and software-defined. Smart environment is an emerging concept that has the potential to address many of the humanity problems. In this context, what are the promising tools for building the smart environments of the future? This paper provides a review of platforms, middleware, and frameworks that can help in this big challenge, discussing their architectures, service life-cycling, digital twins, cloud-based operation, virtualization, security, privacy, communication model, support for AI and machine learning, among other aspects. The proposed revision innovates by employing previous work on future Internet key enablers as parameters for qualitative comparisons. The idea is to determine the degree of alignment among current initiatives for smart environments and the ones emerging from future Internet research. Among the main conclusions are: (i) heterogeneity come to stay; (ii) many contemporary proposals do not cover important aspects raised in future Internet research; (iii) publish/subscribe model is largely employed; (iv) many proposals are stuck to the limitations of current Internet model (another reason to explore the relationship among current platforms and previous future Internet research for IoT); (v) devices interoperability is a problem solved; (vi) ingredients from future Internet research, such as SDN/NFV, ICN and SCN are systematically being adopted for smart environments design. Many others are to come; (vii) AI, machine learning, and big data support are missing not only in TCP/IP-based approaches, but also in future Internet-based.

INDEX TERMS Internet of Things, middleware, platform virtualization, wireless sensor networks, clouds, information-centric networking.

I. INTRODUCTION

Two billion people around the world already use the Internet to read e-mail, interact in the social networks, surf the Web and several other applications. The wide employment

The associate editor coordinating the review of this manuscript and approving it for publication was Peng-Yong Kong.

of the Internet has opened space to aggregate new information sources: objects and machines also can be connected to this big data network. In this context, the Internet of things (IoT) is being defined as to connect all things to the Internet, including aspects as connectivity, data handling, and services [1]. Another technology in the current panorama of IoT is cloud computing, which is a collection of computers

dynamically grouped in servers to provide virtual computation resources to any consumer [2]. Cloud computing is providing the infrastructure for IoT information processing and services. More recently, a debate about placement of computational resources in smart cities has emerged [3]. Fog or edge computing paradigm is being employed when delays are too big due to far distant data centers.

When considering the urban topic, cities are at 2% of the earth surface and already consume 75% of the world available resources [4]. The mashup of both technological paradigms, IoT and cloud computing, is an option for efficient applications development while managing cities resources, a perspective called *smart cities* [5]. Since 2012, 35 projects of North Americans smart cities and 47 European are in progress. They look for solving problems related to the energy, traffic congestion, inadequate urban infrastructure, and others [6].

Beyond cities, the Internet of things promises to revolutionize various sectors of the economy, ranging from Industry 4.0 to rural areas. Virtually, any physical space can become smart – what is being called a *smart environment*. A smart environment is a physical space that employs technology to connect things to the virtual world, aiming to increase the level of awareness of what happens in physical environments and with people (using randomized information). It offers everything as a service (XaaS) by means of digital twins, allowing data analytics to determine trends and reversing changes of the virtual world back to the physical by means of IoT actuators. It integrates IoT, software platforms/middleware/firmware, services, artificial intelligence (AI), machine learning (ML), big data, cloud computing, heterogeneous connectivity, virtual/mixed realities, gamification, and hundreds of other technologies to improve people's quality of life, reducing environmental impact, optimizing the use of physical resources. Smart environments add value to people, businesses, improving sustainability, making efficient and effective use of resources.

A huge set of requirements need to be observed when designing smart environments (or smart cities, as a especial case) [1-80]. Due to the convergence of so many technologies and non-technical aspects, designing smart environments is a quite big challenge. It has been the subject of research not only with current Internet technologies (and cloud computing), but also with new architectures, especially the ones behind future Internet (FI) research [1], [7]–[16]. In this context, future Internet can be defined as alternative architectures for the Internet. Given that the current Internet protocol stack is part of the majority of smart environment proposals and their software platforms/middleware/firmware today, to explore contributions coming from these novel designs (specially from future Internet research) will certainly help on designing the smart environments of the future. This paper aims to contribute to this effort by relating both research paths in a survey of proposals, which are compared accordingly to a common set of aspects.

When compared to the previous work on surveying smart environment proposals, this article offers a unique perspective on the extent to which key future Internet ingredients are adopted in contemporary smart environment technologies. It also includes some proposals originated from FI research to contrast with popular solutions based on current Internet technologies. As it will be discussed in Section III, previous surveys covered specific aspects of proposals (middleware, connectivity, design choices, protocols, etc.). This survey covers the degree of support initiatives offer for smart environment requirements, problems, and limitations. The research questions behind this survey are: (i) *how can future Internet contribute in a better design of smart environments?* (ii) *how smart environments (from current and future Internet) do compare each other from the standpoint of a common set of aspects?*

Given that future Internet proposals aim to solve problems of today's Internet, this article reviews the gaps of current smart environment technology from a FI research point of view. Aspects include dynamic composition of IoT services, naming and name resolution, interoperability of data and architectural entities, smart objects (digital twins) and their relation to services, cloud-based operation, data forwarding and routing, support for artificial intelligence (AI) and machine learning (ML), and communication models as will be explored in Section IV. Therefore, the main contributions of this article can be summarized as follows:

- It provides an unprecedented analysis of existing architectures, platforms, and middleware for smart environments from the unique perspective of FI requirements.
- It presents and discusses previous smart environments' survey articles, pointing their scopes, methodologies, comparison metrics, contributions, and missing aspects. Previous surveys on FI for smart environments are also explored.
- A novel set of key aspects for qualitative comparison of smart environments is provided.
- A detailed comparison of twenty approaches is given considering this mixed set of key aspects, including some proposals from FI research. Open research issues are also presented, accordingly.

The remaining of the paper is organized as follows. Section II defines future Internet, presents its main ingredients, and clarifies its relationships to smart environment design. Section III summarizes related surveys, pointing the existing gaps in literature and the contributions of this article. Section IV provides a brief description of the methodology employed in this paper. Section V presents a number of platforms/middleware, offering a discussion in terms of their architectures, cloud-based operation, communication model, services life-cycling, AI and data analytics, representation of physical devices, security, privacy and trust. Section VI provides a summary and comparison of proposals discussed in the manuscript. Finally, Section VII concludes the paper.

II. FUTURE INTERNET RESEARCH

Future Internet (FI) can be defined as ''any Internet-like net-work that could emerge in the future'' [1]. Or, alternatively, novel architectures and protocols for post-IP Internet stacks. The idea of rethinking the Internet from scratch can be traced back to what is popularly referred as *clean-slate design*. Future Internet research emerged with dozens of initiatives funded by National Science Foundation in the United States, New Generation Network Program in Japan, and 7th Frame Programme in the European Union.

In addition to smart cities/environments and IoT support, FI research integrated many other important ingredients, such as: software-defined networking (SDN) [15], [17], service-centric networking (SCN) [1], [15], information-centric networking (ICN) [1], [12], [15], [18], service-oriented architecture (SOA) [1], [15], self-organizing network (SON) [1], [15], network function virtualiza-tion (NFV) [1], [17], self-verifying naming [15], [18], [19], identifier/locator (ID/Loc) splitting [1], [15], [19], and dis-tributed name resolution. These FI ingredients relate to smart environments requirements and help evaluating the gaps among platforms and middlewares developed with TCP/IP to the ones developed with new approaches. As an example, the ICN research group (ICNRG) of the Internet research task force (IRTF) is thoroughly investigating the relationship of ICN with IoT and smart environments.

Previous work in smart cities goes back to the European future Internet assembly created in 2008 [12]. From 2009 up to 2013, research on smart environment/city related issues has been performed by many EU FI projects, correlated by future Internet assembly working groups, and published in a series of FI books [7]–[11]. This effort has produced a number of requirements and key aspects to be supported by smart environment platforms, middleware, and frameworks from the future Internet research point of view: (i) components architecture and their communication model; (ii) naming, name resolution, and unique global identification; (iii) data contextualization, context-based information delivery, and semantic interoperability; (iv) dynamic service exposition, discovery, contracting, composition, etc.; (v) entities repre-sentation in software layer; (vi) security, privacy and trust network formation; (vii) heterogeneity, resilience, manage-ability; (viii) name-based forwarding and routing; (ix) iden-tifier and locator splitting; (x) software-defined control and operation; (xi) in-network caching; (xii) build-in security; (xiii) self-verifying naming for security; (xiv) control of IoT sensors/actuators and gateways; among many others. Note that these issues were raised considering smart cities as an important application of future Internet [13].

III. RELATED SURVEY PAPERS

The discussion on previous surveys is divided into two por-tions: (i) proposals that adopt the current host-centric net-working technologies; (ii) initiatives that employ other FI paradigms, including ICN, SCN, SDN/NFV, SOA, etc.

A. HOST CENTRIC APPROACHES

In 2012, Miorandi *et al.* [14] covered technologies, appli-cations, and open challenges to realize IoT with current host-centric technologies. A historical perspective has been given, covering from devices up to services and applica-tions. In 2013, Sheng *et al.* [20] provided a survey on IETF protocols for IoT, covering IEEE 802.15.4, 6Low-PAN, RPL, and CoAP. However, discussed topics were restricted to connectivity of nodes and gateways. In 2014, Zanella *et al.* [21] covered technologies, protocols and architecture for Padova smart city. The discussion is more broad, going from sensors to applications, but compari-son to other approaches is very limited. Also in 2014, Fortino *et al.* [22] provided a comprehensive overview of middleware for smart objects (a software that represents a thing) and environments (or places). Middleware are com-pared according to: (i) abstractions they provide; (ii) support for heterogeneity; (iii) management, flexibility, extendability and evolution of smart objects. In 2016, Razzaque *et al.* [23] provided a survey of middleware for IoT. A detailed analysis of requirements has been provided. Middleware have been grouped according to their design approaches, i.e. event-driven, service-oriented, virtual-machine-based, agent-based, tuple-spaces, database-oriented and application specific.

Cruz *et al.* [24] analyzed a reference architecture model for IoT middleware. The differences between the current Internet stack and IoT protocols have been explored, discussing chal-lenges and open issues for IoT middleware. Requirements for IoT platforms have been classified as functional (resource discovery and management, data and event management) and non-functional (scalability, timeliness, reliability, availabil-ity, security, privacy, simplicity, interoperability, flexibility, etc.). Third three IoT platforms have been categorized accord-ingly to their support for: device management, application development, and application enablement.

Guth *et al.* [25] firstly presented a reference architec-ture for IoT platforms and then contrasted eight platforms accordingly to this reference. The comparison included IoT connectivity for sensors/actuators and gateways, IoT integra-tion middleware, and upper level applications. Commercial platforms have been also included.

In [26], Cruz *et al.* provided qualitative and quantita-tive metrics to evaluate IoT middleware performance in a real scenario. Proposed qualitative metrics included: (i) per device authentication; (ii) data access control; (iii) device credentials; (iv) device habits and addresses; (v) develop-ment kits; (vi) supported application protocols; (vii) popu-larity; (viii) number of updates; and (ix) mobile application. Qualitative network performance metrics have been adopted in experiments, i.e. packet sizes, error percentage, response times, etc. Eleven IoT middleware have been qualitatively compared. Five middleware have been installed and con-sidered in experiments. A list of lessons learned has been provided by the authors.

Sethi and Sarangi [27] provided a taxonomy of IoT technologies, including architectures, protocols, and applications. The discussion covered: sensors and actuators, technologies for data preprocessing, IoT connectivity, middleware, and applications. The authors presented a list of open challenges related to IoT middleware: (i) network, semantic, and syntactic interoperability; (ii) programming abstractions; (iii) device discovery and management; (iv) scalability; (v) big data and analytics; (vi) security and privacy; (vii) cloud services; and (viii) context detection. Ngu *et al.* [28] provided an IoT middleware categorization, a comparative analysis of middleware, and a discussion about open research issues. IoT middleware are classified into three types: service-based, cloud-based, and actor-based. Service-based adopts principles of a service-oriented architecture (SOA). Cloud-based employs elastic computing at edge/cloud. Actor-based middleware enable exposition of IoT devices as reusable actors, i.e. using software representatives. Eight IoT middleware are compared considering: (i) device's abstraction in software; (ii) IoT connectivity; (iii) service composition; (iv) monitoring and visualization of results; (v) service discovery; (vi) security and privacy; (vii) data storage; (viii) data stream processing.

Farahzadi *et al.* [29] discussed about IoT middleware features, architectural styles, and services provided. The focus in on approaches aligned to cloud computing, a.k.a. cloud of things (CoT) approaches. Desirable features included: flexibility, transparency, context management, interoperability, reusability, portability, maintainability, resource discovery, trustworthiness, adaptability, security and privacy, and IoT connectivity. IoT middleware's architecture has been categorized as: distributed, component-based, service-based, node-based, centralized, and client-server (web-based model). Twenty middleware have been analyzed accordingly to its architecture, main application, business model, and cloud-based execution. Finally, open issues in the design of CoT middleware have been explored.

Santana *et al.* [30] surveyed software platforms for smart cities. The article focused in answering the question: "What characteristics should software platforms provide for enabling the construction of scalable integrated smart city applications?" Smart city platforms have been categorized accordingly to their main focus: IoT, big data, cyber-physical system (CPS), and general. Twenty three platforms have been evaluated accordingly to their support for: (i) data management; (ii) application management; (iii) IoT connectivity management; (iv) data processing; (v) external data access; (vi) service management; (vii) software engineering tools; (viii) definition of city model; (ix) interoperability; (x) scalability; (xi) security; (xii) privacy; (xiii) context awareness; (xiv) adaptation; (xv) extensibility; and (xvi) configurability. Open research issues have been pointed in terms of privacy, data management, heterogeneity support, energy management, connectivity, scalability, security, and platform maintenance. Finally, a reference architecture for smart city platforms has been proposed.

Hejazi *et al.* [31] provided a survey of IoT platforms, discussing their architectures and components. The aim has been to help users selecting an adequate IoT platform for their projects. The article presented IoT platforms components, protocols, and roles. According to the authors of [31], the aspects that should be considered when choosing an IoT platform are: stability; scalability, flexibility, pricing model, and business case. Twenty one IoT platforms have been compared accordingly to their support for: (i) device management; (ii) communication model among components; (iii) security; (iv) protocols for data collection; (v) data analytics; and (vi) results visualization.

Nitti *et al.* [32] explored the role of virtual objects (smart objects) in IoT. The relation among IoT challenges and virtualization has been established in terms of: (i) semantic description; (ii) addressing and naming; (iii) search and discovery; (iv) context awareness; (v) situation-awareness; (vi) decision making; (vii) mobility; (viii) security; (ix) association between physical/virtual components. Six IoT architectures have been compared regarding these enablers. Context-awareness for IoT has been also discussed in article [33]. Perera *et al.* reviewed eleven IoT middleware according to their support for: (i) device management; (ii) interoperability; (iii) platform portability; (iv) context-awareness; v) security and privacy. Fifty approaches have been evaluated accordingly to context-awareness life-cycling, including context acquisition, modeling, reasoning, and distribution. In [34], Sezer *et al.* surveyed context-awareness for IoT. The authors also included machine learning and big data aspects in their review. The aim has been to relate context-awareness, inference from context, context reasoning, and learning algorithms. The support for device management, data management, real-time analytics, big data analytics, and learning tools have been explored in thirty six IoT platforms.

From standardization perspective, Gazis *et al.* [35] surveyed the major efforts in the scope of IoT, providing a panorama of initiatives and their relationships. The description covered architecture standardization from international telecommunication union (ITU), European telecommunication standards institute (ETSI), oneM2M, telecommunications industry association (TIA), and the alliance for telecommunications industry solutions (ATIS). Standardization of IoT connectivity has been also addressed, including 3rd generation partnership project (3GPP), 3GPP2, Internet engineering task force (IETF), and institute of electrical and electronics engineers (IEEE). Standards for data, support and other aspects have also been covered. A detailed comparison has been provided regarding standards: (i) maturity; (ii) layers covered; (iii) horizontal segments covered; (iv) arrangement (centralized or distributed); (v) domain coverage; (vi) style (SOA or RESTful); (vii) audience; and (viii) prototype availability.

Previous reviews for smart environments partially cover identified requirements, generally leaving out many points previously identified in FI research. For instance,

the following aspects are usually left behind: naming, name resolution, in-network cache, name-based routing, network programmability, network function virtualization, self-organizing networking, content naming, identifier/locator splitting, mobility, entities representatives, service orchestration, dynamic service compose ability, self-verifying naming, etc. In this context, a more generalist approach covering how TCP/IP-based proposals for smart environments relate to the key aspects employed in state-of-the-art future Internet research is missing in literature. In the next subsection, surveys about future Internet-based proposals for smart environments are discussed.

B. FUTURE INTERNET APPROACHES

In 2009, Stuckmann and Zimmermann [12] commented on IoT as a component of FIAs, but did not describe existing approaches at that time. This has been latter performed by Hernandez-Munoz *et al.* [13] in 2011, covering general features required by smart cities and presenting SmartSantander approach.

The majority of survey papers on future Internet architectures (FIAs) relates to ICN-based IoT (ICN-IoT), therefore focusing on the integration of two FIA ingredients: IoT and ICN [12], [18]. ICN can be defined as a network in which contents are discovered and retrieved using their data names instead of host addresses. Content names are used in packet headers instead of locators. However, future Internet research is not limited to ICN. There are many other components to be integrated, such as: SOA and XaaS, SCN, SDN/NFV, ID/Loc splitting, cloud-computing, contract-based operation, dynamic service composition, recursive layering, autonomic computing, cognitive radio, among others. Despite the importance of these other themes, there are few surveys on ingredients other than ICN and SDN.

The literature has many papers comparing named-data networking (NDN[1]) [18] and MobilityFirst[2] regarding IoT support. These comparisons have been done in terms of: (i) device's discovery; (ii) communication model; (iii) mobility; (iv) naming and name resolution; (v) security; (vi) in-network caching; (vii) scalability; (viii) quality of service; (ix) energy efficiency; (x) support for heterogeneity; (xi) content discovery and delivery; and (xii) data morphing.

ICN naming and name resolution for IoT is a hot topic. The adequacy of self-verifying naming for ICN-IoT is also being investigated [1]. Self-verifying names (SVNes) are flat names generated by hash functions. Hierarchical and flat (SVNes) naming approaches are being adopted for smart homes, vehicular networks, smart buildings, under water communications, and smart campus. In addition, hybrid naming schemes are being investigated.

Regarding packet forwarding and routing, ICN-IoT demands not only name-based routing (NBR), but also look-up-based resolution system (LRS). While NBR is adopted by

[1]https://named-data.net/
[2]http://mobilityfirst.cs.umass.edu/

NDN (Subsection V-O), MobilityFirst employs LRS (Subsection V-N). Both techniques are required for ICN-IoT data discovery and delivery, since each of them has important advantages for specific IoT scenarios. Also, the identification of services and content is a requirement for ICN-based smart environments. Efficient name aggregation, i.e. hierarchical naming structures, is also a requirement being addresses by FIAs.

The work on NDN [18] for IoT encompasses naming, name resolution, forwarding and routing, access control and policies, in-network caching, device configuration and management, data aggregation, available libraries for platform/middleware development, operating systems. NDN is being adopted for: cyber physical systems, environment monitoring, smart home, smart building, vehicular networks, smart healthcare and smart cities. This shows that there is a strong relationship between smart environment and FIA design.

Several wireless sensor networks are now employing SDN and virtualization to provide sensing-as-a-service. The integration of SDN/NFV and cloud computing help on meeting smart cities' requirements for flexibility and elasticity of network functions and physical resources. Taleb *et al.* [17] addressed IoT relation to SDN and NFV, specially regarding security features. SDN and NFV have gained increasing attention to react to many IoT security threads. Twelve SDN/NFV-based approaches for IoT security have been compared accordingly to their support for: (i) policy-based orchestration; (ii) intrusion detection; (iii) autonomic reaction; (iv) SDN-based; (v) cloud-based; and (vi) ETSI NFV-based. A list of lessons learned (in contrast to the traditional IoT security) is provided, as well as the open research issues required for better integration of IoT, SDN, and NFV. SDN-based smart cities are also being explored in literature.

IV. METHODOLOGY

The key aspects selected for discussing and comparing smart environment proposals and their platforms and middleware have been obtained from previous work in smart environments performed in the context of future Internet [1], [7]–[16]. Since, the amount of relevant aspects is enormous and it is virtually impossible to include all the relevant ones, the approach taken here has been to select the requirements addressed in two previous work from Alberti *et al.* [1], [19]. They are related to the NovaGenesis future Internet architecture, which has been designed to support IoT and smart environment needs since 2008. NovaGenesis [1], [16], [19], [36], NDN [18], and Mobility-First already summarized many of the requirements for future smart environments and therefore some of its key aspects are representative for this task. They are:

- Architecture - It covers how proposals are structured and how their components interact with one another.
- Cloud Computing - The cloud computing model is advantageous to provide flexibility and operation cost

reduction of computing systems. Intelligent environment solutions need to support distributed execution across regional and global data centers. Edge computing is also interesting for smart environments and it is explored by some proposals.

- Communication Model - It discusses how hardware and software components communicate. IoT connectivity is one of the most important topics in any smart place proposal. Typically, traditional client/server models are employed in CoAP and REST. Notwithstanding, publish/subscribe model is preferred for other proposals, e.g. MQTT. Finally, future Internet architectures bring new models for smart environments, such as push/pull.

- Naming, Name Resolution, and Semantic Interoperability - The role of naming and name resolution is better discussed in future Internet architectures, since naming is fundamental for smart environments. IoT demands for semantic interoperability of names, contents, services, etc. Since all smart environment' entities have names and Internet's domain name system (DNS) provides limited name resolution, novel naming and name resolution proposals are emerging such as: Global Name Resolution Service (GNRS) from MobilityFirst architecture, NDN DNS (NDNS) and NRNCS [16], [19]. They provide a powerful tool for supporting security, privacy and trust (SPT) [18], as well as service compose ability [1], [16].

- Service Life-cycling - It addresses IoT services and their life-cycle, including dynamic compose ability and service contracting. IoT services have the ability to add value to business and therefore must be present in any analysis of smart environment proposals. The integration of SDN and NFV brings a dynamism for service orchestration and network programmability. Smart environment architectures are adopting these technologies that emerged in future Internet research. Novel approaches for service-oriented architecture are also being combined with SDN/NFV, such as ICN-IoT and contract-based operation [19].

- Artificial Intelligence - Autonomic computing, machine learning (ML) and artificial intelligence (AI) are fundamental ingredients to automate environments, making them truly more autonomous and smart.

- Entities Representation - Proposals should address interoperability of data, things, gateways, and services, since heterogeneity of hardware and software is the rule. A candidate solution is to employ entities representation services, i.e. smart objects or digital twins, to represent heterogeneous entities via a common interface.

- Security, Privacy and Trust (SPT) - It is a consensus that SPT requirements should be met by any smart environment proposal. Name-based security is emerging as a novel paradigm, in which data chunks integrity is granted by their own names.

V. SMART ENVIRONMENT PROPOSALS

Smart environment proposals usually involve the integration of platforms, middleware, and/or frameworks with IoT devices. In this context, a platform is a major piece of software that other applications operate and run over. A middleware is a mediator employed to transfer data from heterogeneous protocols, platforms, and operating systems (OSs) to applications. In the context of IoT, a platform for smart environment can include one or more middleware. Finally, a framework can be defined as a collection of libraries, which can be used to develop platforms and middleware. A framework provides higher level features to allow developers to focus on application logic.

In this section, the aforementioned key aspects have been analyzed for many proposals (TCP/IP and future Internet research), which have been selected based on availability of high quality publications and coverage of selected aspects. In addition, commercial platforms whose details are not available by trustable means, specially by peer reviewed papers have been avoided. Proposals that are unclear or that do not provide consistent information with regard to the selected key aspects have been disregarded. Please note that the list is not complete as it is impractical to review all available options.

A. ALLJOYN

AllJoyn [37] is an open source platform that emerged for peer-to-peer, device-to-device communication in smart home, building, industry and city context. It provides a set of protocols and services for interoperability among devices and software applications in smart environment. Originally, the scope was dynamic proximal networks, i.e. proximity communication among devices in the same environment. However, the need to involve software applications in the cloud has forced the proposal to expand by providing gateway agents and device system bridge. AllJoyn has been merged with another platform called IoTivity (Subsection V-J), both being developed by Linux Foundation.

1) ARCHITECTURE AND CLOUD COMPUTING

AllJoyn adopts a decentralized architecture in which compatible devices can host a router and one or more services and applications. This architecture contains four layers: transport layer, router/client connectivity layer, service layer, and application layer. The router/client connectivity layer contains a re-implementation of the D-Bus specification, including support for decentralized devices. Therefore, AllJoyn relays in multiple underlying connectivity provided by a transport layer that supports TCP, UDP, and even Unix sockets inside OSs. These inter node protocols are carried over Wi-Fi, Ethernet, Bluetooth, and power line communications. The router/client connectivity layer also offers service and interface discovery, connection establishment, and security independently of OSs. Every node can have its own router for D-Bus-based message exchanging. A lightweight node

has been also developed for low power devices that want to join the network. In this case, the majority of AllJoyn functionalities is provided by a gateway node. The service layer contains functionalities for devices onboarding, control panel, notifications, audio support, among others. AllJoyn was not originally created to operate in cloud. However, this need appeared when smart phones and other remote devices become required in application scenarios. Thus, a gateway agent and a device system bridge have emerged to deal with devices and systems connected via cloud computing. For instance, Masek *et al.* [38] applied AllJoyn in a light bulbs scenario integrated to the cloud to enable interoperability with a Twitter application running in a smartphone.

2) COMMUNICATION MODEL
AllJoyn services and applications inside a node are built by objects that have more than one interface to support three types of communication: (i) remote method call; (ii) signals; and (iii) properties. An application interested in a method implemented in another object must access this method through a proxy object locally. The call passes the expected input parameters and wait for the results. A client/server model is used in method calls. Signals are notifications provided by one service to others. They communicate events or state changes. Connection oriented sessions deliver a signal to all nodes connected. Meanwhile, sessionless signals are delivered to nodes that subscribed for them, i.e. a pub/sub model is employed. Routers maintain a cache to store versioned versions of sessionless signals.

3) NAMING, NAME RESOLUTION, AND SEMANTICS INTEROPERABILITY
Naming has an important role in AllJoyn. Naming structures proposed in D-Bus specification are adopted and extended. Naming covers application, service, interface, interface members (methods, signals and properties), and well-known names. Applications need to connect to a single router, which assigns a unique name for them. A next-generation name service has been developed to enhance discovery feature. It uses multicast DNS-enabled discovery of nodes and services that contain a certain interface.

4) SERVICES LIFE-CYCLING AND ANALYTICS/AI
AllJoyn supports two types of discovery: name-based and announcement-based. Name-based discovery enables applications/services to advertise and discovery peers. A discovery protocol is implemented in routers that exchange discovery messages via multicast IP. The well-known names an application supports are advertised in a message generated by its router to other proximal nodes. Some interested application (in another node) can ask its local router about such well-known names. The local router then sends a service discovery notification to the peer application informing about the application that advertised before. A connection is mounted to carry out the partnership. In announcement-based discovery, advertisement and peer discovery occur based on

interface names. Applications broadcast announcement messages that they have interest in receiving notifications about certain interfaces in AllJoyn routers. When a consumer node is closer to a router that registered previous broadcast announcements, the router delivers the notifications. Device discovery is supported at the AllJoyn service layer. The support for analytics, AI, and machine learning is outside the scope of AllJoyn platform. Villari *et al.* [39] provided a Big data integration of AllJoyn with MongoDB and Storm to deploy a smart home scenario.

5) ENTITY REPRESENTATION
In remote method calls, a proxy object represents a producer service object inside a consumer device. This representation is aimed for remote process call only. Therefore, it has no relation to device representation in the format of a digital twin.

6) SECURITY, PRIVACY AND TRUST
AllJoyn router/client connectivity layer provides the typical mechanisms for authentication, authorization, secrecy and integrity. Interfaces in application layer can be annotated as secure. In addition, connections are also secured by symmetric or asymmetric cryptography. A Security 2.0 package includes a security manager application for establishment of security mechanisms among producers and consumers. A permission module is added to AllJoyn core enabling credentials exchange. Also, the support for policies, certificates and trust anchors has been added. Applications are identified using a combination of identifiers (global unique identifier - GUID) and identity certificate.

B. ALMANAC
ALMANAC [40] is a FP7 funded project focused on providing a service platform for smart cities. The smart city platform (SCP) supports semantic interoperability of sensors, services and data management. Also, it enables federation of IoT networks, covering policy definition, interoperability to external applications, and intelligent operation of city devices and services.

1) ARCHITECTURE AND CLOUD COMPUTING
The ALMANAC architecture aligns to IoT-A reference architecture [41] and addresses the following main topics: (i) exposition of SCP services and devices; (ii) data context management; (iii) translation of specific data representations and communication protocols. The architecture is organized into four layers:

- Smart city resource adaptation layer (SCRAL), which provides a uniform REST API to physical devices via a field access layer (FAL). The FAL has a collection of drivers for IoT technologies, like ZigBee, Xilevy, etc. It also includes policy enforcement to provide access control and role-based policies.
- Data management framework (DMF) enables storage, filtering, aggregation, fusion, retrieval and management

of data and metadata. It includes four internal components: data fusion manager (DFM), storage manager (SM), resource catalogue and semantic representation framework (SRF).

- Virtualization layer (VL) provides support for interoperability to other platforms, data security and access policies enforcement. It includes three internal components: virtualization layer core (VLC), LinkSmart and federation identity and access manager (FIAM). LinkSmart is a European middleware for interoperability of platforms, enabling ALMANAC to interoperate to other proposals. Therefore, cloud-based operation is native.
- API layer provides REST and web socket APIs to offer user access to the platform.

2) COMMUNICATION MODEL
SCRAL supports pub/sub (MQTT) and client/server (HTTP) communication models. REST API is employed for request/response interactions, while web socket is used for data stream. Many ALMANAC functions are exposed via SCRAL.

3) NAMING, NAME RESOLUTION, AND SEMANTICS INTEROPERABILITY
ALMANAC employs DNS (CNAMEs) to provide component naming and name resolution. A name structure is employed for intra and inter platform instances (PIes) communication. Data validation is performed at SCRAL as well as metadata generation. Data aggregation and filtering are provided by DMF. Heterogeneous device features are uniformed, abstracted, and translated to a semantic representation framework (SRF). SRF provides metadata and context based on semantic web standards (JSON-LD, OWL). It also enables SPARQL queries. VLC also translates request/response messages to allow interoperability.

4) SERVICES LIFE-CYCLING AND ANALYTICS/AI
ALMANAC provides federation of PIes, enabling internal and external components (services) to be integrated to the set of available services. LinkSmart solution enables data and task exchange among PIes. Service discovery is based on DNS. AI, machine learning and analytics are not integrated to the platform. In [42], Bonino *et al.* provide an application example of ALMANAC to waste recycling. Cloud APIs are employed to deliver the location of waste bins and their garbage level to an Android OS smartphone application. The services employed in this solution have been described and evaluated in a real scenario in Turin city, Italy.

5) ENTITY REPRESENTATION
The resource catalogue handles descriptions of physical devices, enabling querying of available resources either by REST explicit registration or by implicit UPnP service control point document (SCPD). The update of devices states is at DMF layer. DFM offers complex event and live data processing through Esper, QSO2 and StreamInsight tools.

In addition, VLC acts as proxy for lower layers, forwarding data to/from the right modules locally or at federated peers. It also supports complex application requests, such as data fusion query or permanent data query.

6) SECURITY, PRIVACY AND TRUST
FIAM takes care of security and trust, enabling federation of ALMANAC platform instances or interoperability with other platforms, like SmartSantander. FIAM centralizes users' authentication and authorization. LinkSmart also enforces access control and security.

C. ANEKA
Aneka is a platform-as-a-service (PaaS) proposal developed by Manjrasoft company [43]. It provides an IoT middleware that interfaces to public and private cloud APIs, more specifically Microsoft Azure, Amazon EC2 and GoGrid. It provides data stream reading directly from sensors or from databases. Also, data analytics is offered to process data at the cloud. When interest events are detected, outcomes are sent to a visualization tool. Aneka encompasses a set of APIs for developers and several services that allow users to configure, scale, reserve, monitor and bill IoT resources used by their applications.

1) ARCHITECTURE AND CLOUD COMPUTING
Aneka architecture is supported by a cloud-based infrastructure. Sensors' data are received by gateways and forwarded to public or private cloud storage. Data access security is granted by cloud providers. The Aneka software runs in the same cloud than sensor's data storage or any other cloud. It consists of a number of services for IoT scenario: (i) QoS-based, dynamic scheduling of computational, storage and networking resources; (ii) flexible billing; (iii) cloud web portal for analytics and visualization service; and (iv) backup cloud storage of sensor data. Optimal virtualized resources are allocated or terminated according to the requirements of each sensor application in the platform. This is accomplished by dynamically negotiating with cloud IaaS providers according to application history and budget. Subhash *et al.* [44] presented a solution for elasticity of cloud resources for Aneka platform.

2) COMMUNICATION MODEL
Aneka follows the typical communication model offered by cloud computing providers. For instance, Azure IoT hub offers pub/sub (MQTT and advanced message queuing protocol - AMQP) and client/server (HTTP).

3) NAMING, NAME RESOLUTION, AND SEMANTICS INTEROPERABILITY
The initiative does not enter in this scope. Data semantic interoperability is provided by IoT services in cloud providers.

4) SERVICES LIFE-CYCLING AND ANALYTICS/AI
Aneka proposal is centered on elasticity of cloud resources for IoT applications. An scheduler selects and assigns

resources accordingly to IoT applications requirements. It dynamically negotiates resources with public cloud providers to meet IoT application requirements. Service life-cycling depends on the cloud support. MapReduce data analytics is supported. No mention to AI technologies.

5) ENTITY REPRESENTATION

Once again, Aneka delegates a functionality for cloud providers.

6) SECURITY, PRIVACY AND TRUST

Aneka assumes commercial IaaS suppliers security and privacy mechanisms are sufficient for IoT scenarios.

D. ARROWHEAD

It is lightweight framework aimed at enabling the Industrial Internet of Things and to improve interoperability between applications [45]. The framework operates with a hierarchical set of core systems, allowing a single machine to operate its own Arrowhead cloud. Local authorization and orchestration rules are supported. Arrowhead provides inter-cloud services and an approach to solve security and orchestration issues outside the cloud limits. It aims to guarantee the interoperability of different technologies through the advantages of SOA. A core system is proposed accordingly to a new concept: the Gatekeeper. Gatekeepers are special gateways with the extra functionality of decision-making instead of just network layer functions.

1) ARCHITECTURE AND CLOUD COMPUTING

Arrowhead is composed by systems that interoperate with services. The framework has been designed to run in local or online clouds. Three obligatory services are: (i) service registry management; (ii) service authorization; and (iii) service orchestration. Other services can be added at each local cloud. Services can interoperate globally by using gatekeeper systems that search for resources in other domain clouds. Therefore, it offers resources for global service discovery and inter-cloud negotiation.

2) COMMUNICATION MODEL

The proposal applies technologies such as 6LowPAN, REST, CoAP, XMPP, ZigBee and OPC-UA. Therefore, different communication models are supported. Jokinen *et al.* [46] presented two IoT application for smart cities: (i) a street lighting system connected via ZigBee; and (ii) a car heating system for winter connected using 6LowPAN. The integration of both systems using Arrowhead platform has been described in details.

3) NAMING, NAME RESOLUTION, AND SEMANTICS INTEROPERABILITY

Arrowhead prescribes a method of documentation in order to enable native interoperability among services. This mechanism employs different semantics and communication protocols [47]. Gatekeepers encompass addressing translations.

Derhamy *et al.* [47] do not offered real names on paper examples, just fictional ones. Therefore, it is not possible to evaluate its naming approach.

4) SERVICES LIFE-CYCLING AND ANALYTICS/AI

Arrowhead is based on service oriented architecture (SOA) design. The focus is on global interoperability of IoT services among cross domain SOA platforms. It supports SOA-based design principles, including: (i) standardized service contracts; (ii) service loose coupling; (iii) service abstraction and (iv) service autonomy. The Orchestration system allows dynamic reconfiguration of service consumer and provider end-points. Inter-cloud service discovery is provided meaning that local clouds can consume outside services or provide data as a service to outside consumers. The Arrowhead framework does not include native tools for analytics or AI. In a more recent paper, Kozma *et al.* [48] presented a supply chain management use case employing Arrowhead framework. A detailed description of the services implemented has been provided.

5) ENTITY REPRESENTATION

The project is focused in IoT automation via cloud services and their interoperability. However, the degree of support for entity representation is not clear.

6) SECURITY, PRIVACY AND TRUST

The proposal offers support for authentication and authorization in a local cloud. For inter-cloud security, Arrowhead offers decentralized certificate management and trust formation among gatekeepers.

E. AMAZON WEB SERVICES IOT

Amazon web services (AWS) provides a commercial cloud-based platform for IoT. It has been launched in October 2015 to provide a complete solution for smart environments. In the edge, the platform covers IoT devices operating system (FreeRTOS) and edge computing (Greengrass) to collect and analyze data in the access network. In the cloud, services offered for IoT integrate to other Amazon products, such as Kinesis (for machine learning), DynamoDB (for data storage), and QuickSight (for visualization).

1) ARCHITECTURE AND CLOUD COMPUTING

Taking a horizontal perspective, AWS IoT is structured from endpoints up to enterprise application in the cloud. Endpoints can be IoT sensors and actuators running MQTT over TCP/IP/Wi-Fi or other link layer protocols, such as power line communication (PLC), Ethernet, Bluetooth low energy (BLE).

Devices of the open platform communications unified architecture (OPC-UA) standard are also supported via protocol adapters at the edge gateway. The gateway can run edge computing services, including: (i) over the air (OTA) firmware update for end devices; (ii) local device state shadowing; (iii) support for device to device (D2D)

communication; (iv) support for intermittent device communication; (v) local execution of predictions based on machine learning; (vi) data filtering; (vii) security related services. The next portion of the architecture (AWS IoT Core) implements a message broker, which provides a rule engine to filter, transform and aggregate device's data. The rule engine can also evaluate, transform, and forward gateway messages to cloud services. All these actions are performed following if-then-else logic. For instance, all the data sent by a connected vehicle using a MQTT topic like 'connected-car/telemetry/#' is processed and forwarded to persistent storage in the cloud. The message broker also includes device shadowing and things' security certificate management. The architecture also includes a component for device management, which supports OTA updates, device and group of devices searching, and batch device provisioning. An IoT device defender is included to audit devices' configurations, to monitor device behavior and mitigate effects of misbehaving nodes. The next component is IoT Analytics, which encompasses data pipelines, storage, in-depth data analysis and reports. Finally, a set of enterprise applications can be connected to offer visualization, machine learning, etc.

2) COMMUNICATION MODEL

From a vertical point of view, MQTT with pub/sub communication model is used from devices up to IoT Core. HTTPS for devices publication is also possible. In the cloud, client/server RESTful HTTP is predominant. Bhatnagar *et al.* [49] has presented a car pollution detection system that relays in AWS for user warning in case of excessive CO_2 levels. MQTT is employed. Another use case for smart vehicle traffic control has been proposed by Tarneberg *et al.* [50]. Traffic lights, buses, induction loops, and bus stops are connected via MQTT topics. AWS IoT provides device management and authentication. Results have been obtained by experiments and simulation.

3) NAMING, NAME RESOLUTION, AND SEMANTICS INTEROPERABILITY

Things, groups of things, type of things, policies, data base tables, services, rules, certificates, edge/cloud functions, authorizers, and IoT data topics are named in natural language. Data topics names are adopted not only for MQTT, but also in URLs for RESTful HTTP. It is not clear the degree of support AWS IoT has for name resolution and semantic data interoperability.

4) SERVICES LIFE-CYCLING AND ANALYTICS/AI

Services can be dynamically invoked in the form of Lambda functions not only in the cloud, but also inside gateways and/or devices in the edge. AWS IoT provides a web-based graphical user interface with thousands of possible interactions. However, a lot of manual intervention is required, as reported by Tarneberg *et al.* [50]. AWS IoT supports integrated data analytics and machine learning.

Amazon QuickSight business intelligence application can also be connected to the IoT platform.

5) ENTITY REPRESENTATION

The best abstraction in AWS IoT for entity representation is the Device Shadow, which is a JSON document containing the latest state information of a certain thing. Device states can be changed by MQTT or HTTP. Named topics are used for this aim.

6) SECURITY, PRIVACY AND TRUST

Security encompasses authentication and secrecy in all connections. It contains SPT mechanisms for devices (FreeRTOS), gateways (Greengrass), cloud (IoT Core), and device management. To connect, devices should have proper credentials (X.509 certificates). All the IoT traffic is ciphered by transport layer security (TLS). AWS IoT also provides a service to continuously monitor and audit device security settings: the Device Defender. The objective is to ensure that best security practices are being fulfilled. If any device is disregarding the best security practices an alert is generated. Device behavior deviations are monitored.

F. CLOUT

ClouT is a partnership composed by European and Japanese companies, universities and research centers [51]. The main object of this consortium is to offer services as an alternative to provide connection among IoT and Internet of people. This approach also offers to end-users the possibility of creating their own cloud services once the platform is user-centric.

1) ARCHITECTURE AND CLOUD COMPUTING

ClouT is based on three main layers: the CIaaS (City Infrastructure as a Service), CPaaS (City Platform as a Service), CsaaS (City Application Software as a Service). The CIaaS includes sensors that are virtualized as a middleware service in the CPaaS. The application software in the CsaaS uses the CPaaS platform and it is the layer open for the end-users. The virtual resources exposition in the CIaaS is guaranteed through open standard APIs. It enables the access to any physical interconnected resources in the CIaaS, empowering anyone/anything to use the devices, data and computational power.

2) COMMUNICATION MODEL

ClouT adopts a multi-protocol IoT gateway for managing and collecting data flow from heterogeneous devices. The IoT Kernel block has 3 components: (i) Uniform Access to IoT Devices; (ii) IoT Device Management; and (iii) IoT Device Traffic Adaptation. Uniform Access to IoT Devices is the component responsible for device abstraction, implementing an API for transparent access to the upper layers using extensible messaging and presence protocol (XMPP). To meet the large volume of sensors and actuators expected in a smart city, a variation of the Openfire server was developed, an open source project that provides a client application for

unstructured messaging and a server that supports group chats employing XMPP [52].

3) NAMING, NAME RESOLUTION, AND SEMANTICS INTEROPERABILITY

Heterogeneous data is mapped to a common data format. Interoperability is treated at CIaaS and CPaaS. Naming and name resolution is based on current Internet.

4) SERVICES LIFE-CYCLING AND ANALYTICS/AI

ClouT offers a City Service Composition module that is formed by two functional components: (i) Service Composition and (ii) Development and Deployment Platform. Service Composition contains all the tools necessary for users, whether experienced developers or simple citizens, to build their own services. Service composition is achieved by employing services exposed by the CIaaS layer using mashup tools. A web application called Service Mashup Editor was created to allow intuitive and GUI-based manual service creation. Development and Deployment Platform offers virtual machines that host ClouT applications. As the number of requests grows, new instances of the hosted application are implemented, promoting load balancing. This component supports a variety of software platforms, such as web servers, SQL/NoSQL databases and middleware, and a management interface for system developers and administrators. Data/event processing and decision making are offered in the CPaaS layer. Few details are given about this component, even in project deliverable.

5) ENTITY REPRESENTATION

The IoT Device Management is responsible for management tasks at IoT nodes, such as parameterizing the frequency used to send data, checking the state of firmware, among others. This component also has the Device Discovery function, which is performed in different ways depending on the standards supported by each device. The discovered devices are stored in a local database. The IoT Device Management enables physical entity representation.

6) SECURITY, PRIVACY AND TRUST

ClouT ensures client authentication through user and password and rule-based access to resources. It covers standard security features, including protocols selection, authorization of access to platform modules and encryption for protection of sensitive data. The security module is composed of 3 components: (i) Platform and Infrastructure Reliability Monitoring; (ii) Cryptography; and (iii) Authentication, Authorization and Auditing.

G. COMPAAS – COOPERATIVE MIDDLEWARE PLATFORM AS A SERVICE

COMPaaS [53] is a platform that covers from heterogeneous physical devices up to smart environment applications. The proposal extends EPCGlobal [54] towards interoperability of RFID and other IoT standards. It is web-based and supports synchronous and asynchronous communication.

1) ARCHITECTURE AND CLOUD COMPUTING

The COMPaaS architecture is divided in three layers: (i) physical devices layer; (ii) event processing and integration layer; (iii) services and applications layer. Physical devices layer encompasses a set of smart objects called *logical resources*. The event processing and integration layer provides a web-based middleware with several services: communication with IoT devices, physical resource management, complex event processing (CEP), devices operation, data collection, administration and query. The application layer takes advantage of the middleware to start data collection cycles for devices. In [53], a proof-of-concept is performed using virtual machines, therefore demonstrating the proposal is ready to run virtualized.

2) COMMUNICATION MODEL

For synchronous queries, COMPaaS employs RESTful APIs among layers. Middleware and logical resources use a Subscribe/Notify communication model implemented with RESTful for device control and data collection. However, the platform also allows asynchronous bottom up communication. An Observer communication model is implemented using simple object access protocol (SOAP) web sockets for asynchronous responses.

3) NAMING, NAME RESOLUTION, AND SEMANTICS INTEROPERABILITY

Logical resources are described by profiles, which contain device names, manufactures, mode, URI, etc. Applications use profiles to discover resources. Name resolution is not covered. Semantic interoperability is addressed by device profiles and DataMessage objects, which represent data generated by smart objects.

4) SERVICES LIFE-CYCLING AND ANALYTICS/AI

Services are static. Nothing is commented on how to add new services for the platform. Therefore, service discovery is not addressed. Data analytics and AI techniques are not covered in the proposal.

5) ENTITY REPRESENTATION

Logical resources are used to represent physical devices inside the platform. They contain drivers for communication with devices (e.g. RFID and WSN).

6) SECURITY, PRIVACY AND TRUST

Security support is not explored in [53]. The platform has an administration service, but no details are given.

H. DIAT

The distributed Internet-like architecture for things (DIAT) [55] is a deliverable of the iCore FP7 funded project.
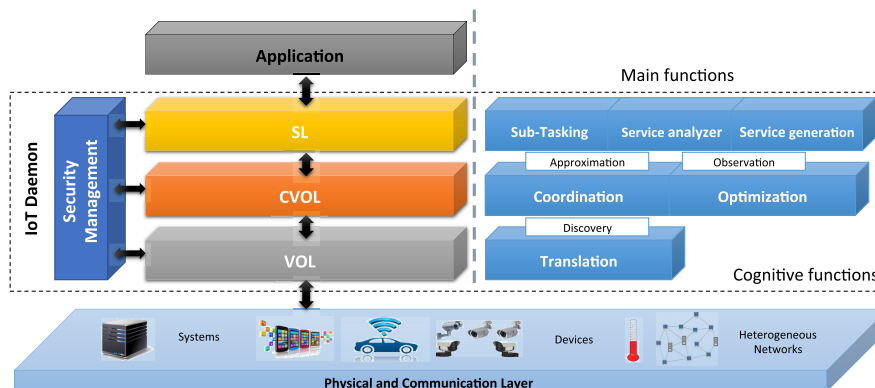
FIGURE 1. Distributed Internet-like architecture for things (DIAT). Physical devices are represented by virtual objects (VOs) that facilitate their dynamic recognition, composition and adaptation. VOs can be aggregated to form a composite virtual object (CVO). CVOs provide tiny services that interoperate with applications.

The main aims are: (i) to deal with heterogeneity of IoT devices; (ii) zero configuration of new devices and services; (iii) self-management of data/services, including autonomic composition; (iv) control policy model for security and privacy. The architecture is focused on autonomous data collection, processing, semantic annotation, decision making with minimal human intervention. In this sense, DIAT addresses situation and context awareness. It provides dynamic recognition, composition, and adaptation of IoT objects, which are virtual representatives of physical devices. Virtual objects (VOs) and their compositions provide tiny services for applications. DIAT employs dynamic service instantiation and modeling to create a distributed environment for smart environments. DIAT supports virtual object services exposition in order to facilitate self-orchestration.

1) ARCHITECTURE AND CLOUD COMPUTING
DIAT is organized into three software layers (as illustrated in Figure 1):

- Virtual object layer (VOL) - It represents physical devices inside DIAT architecture, acting as an interpreter among physical and virtual worlds. It provides a semantic description of devices features and capabilities to enable access to any physical device. A virtual object is equivalent to the aforementioned smart object concept. Virtual objects can run in data centers or computational capable devices.
- Composite virtual object layer (COL) - It enables communication and coordination among virtual objects. A composite virtual object (CVO) is dynamically composed according to the needs of services (tasks). It mash ups VOs and/or other VCOs accordingly to service needs, scheduling actions towards collaboratively accomplishing a service request. CVO formation involves selection of suitable VOs that can "socially" (the so called self-emergent behavior of autonomic systems) solve the task proposed. A discovery mechanism (not proposed by the authors) should discover and select

the appropriate VOs. Semantic description is fundamental on this step.
- Service Layer (SL) - It receives task requests from users and allocates appropriate services to handle them, a feature called automatic service creation. This layer can subdivide a received task into sub tasks. If there is not a perfect match, the service layer can partially meet the request, an approach named *approximate service*.

2) COMMUNICATION MODEL
Many IoT solutions employ MQTT pub/sub for the communication among applications, services and adapters. DIAT is agnostic regarding its communication model. However, apparently, several physical object specific technologies will be required.

3) NAMING, NAME RESOLUTION, AND SEMANTICS INTEROPERABILITY
Naming and name resolution are not seen as a main concern and are based on current Internet technologies. DIAT employs metadata for service discovery and selection in order to accomplish a task. It also employs semantic operators for human location (e.g. "atOffice") and status (e.g. "inMeeting").

4) SERVICES LIFE-CYCLING AND ANALYTICS/AI
DIAT supports autonomic service creation according to required tasks. Service life-cycling considers location-, situation- and user-awareness. Context is derived for VOs continuously by an observer component that is spread over CVO layer and SM. Changes in context can drive initialization of new services, autonomously. Location-awareness is given using semantic operators like "atHome", etc. In DIAT, there is huge dynamism in service life-cycle, while it is more typical in IoT platforms that services remain running for long term.

Self-orchestration is achieved based on context-aware decision making. VCOs mash up VOs and/or other VCOs
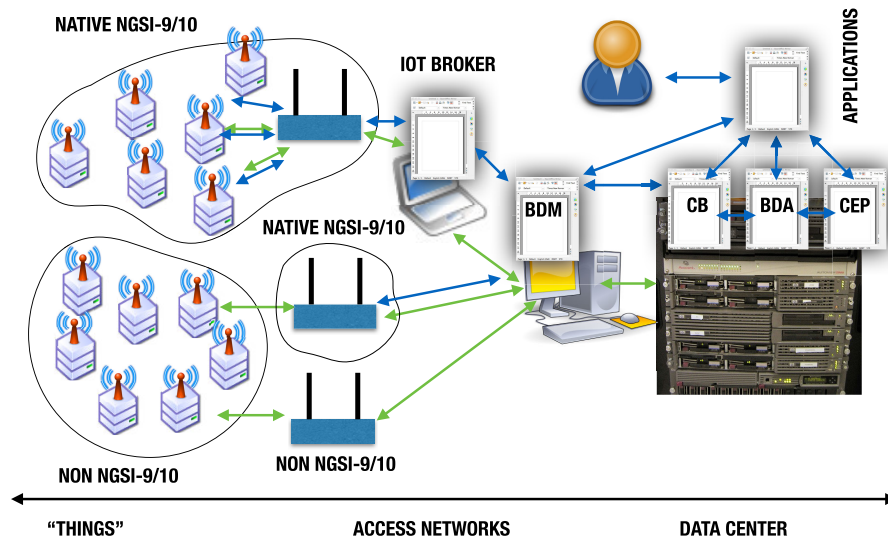
FIGURE 2. FIWARE scenario for converging future Internet, Internet of "things", big data, and cloud computing.

accordingly to service needs, scheduling actions towards collaboratively accomplishing a service request. DIAT receives task requests from users and allocates appropriate services to handle them, a feature called automatic service creation. Besides self-management support, the integration of artificial intelligence and data analytics in the architecture is not clear.

5) ENTITY REPRESENTATION

Both VOL and COL support physical and logical entity representation. Protocol adaptation and data exposition are performed based on these digital twins, a.k.a. virtual objects. Dynamic compositions and semantic-based resource access and control are provided by DIAT's VCOs.

6) SECURITY, PRIVACY AND TRUST

DIAT encompasses a security management (SM) component. It is a cross layer service that employs expressive policies (with semantic) to model, provide and enforce access control to entities. There are representations for data, identities, context, roles, structure and behavior. The SM runs into an IoT *daemon*, which can be simplified for lightweight devices. The collection of meta models and components is called SecKit.

DIAT offers a security management (SM) component that employs expressive policies (with meaning) to model, provide and enforce access control. Security in DIAT starts at access policy management. Policies are modeled using specific ontology tuples, which contains policy-ID, VO's authorization, obligation, role, behavior and conditions under which the policy is applicable. Policy rules are specified following an event-condition-action model.

I. FIWARE

FIWARE [56], [57] is the technological platform of the European FI-PPP initiative. It is an open ecosystem that uses a standardized software platform to facilitate the development of smart applications in various sectors, including the IoT and smart cities. FIWARE platform integrates services using next generation service interfaces (NGSIs) as a glue. The architecture enables new services to be added as *generic enablers* (GEs). GEs offer various functionalities, implementing protocols and interfaces for control and data planes.

1) ARCHITECTURE AND CLOUD COMPUTING

Figure 2 illustrates a simplified overview of FIWARE architectural components for IoT, which include: (i) IoT broker; (ii) backend device management (BDM); (iii) context-broker (CB); (iv) big data analysis (BDA); and (v) complex event processing (CEP). Before describing them, it is important to notice that three kinds of things are supported: (a) devices that are compatible with next generation service interface (NGSI) version 9/10; (b) devices that are not compatible with NGSI 9/10, however the gateways are; and (c) devices and gateways that are not compatible with NGSI-9/10.

The IoT broker recovers, collects and processes information from things exposing devices as RESTful resources [58]. The BDA exposes legacy technologies (standardized or proprietary) as resources to the CB via NGSI-9/10. IoT agents are instantiated to handle, configure and monitor non NGSI devices and gateways. The CB provides a publish/subscribe context broker service via NGSI-9/10 interface. Contexts can be registered, updated, queried, notified, subscribed, etc. For example, a native NGSI-9/10 device can create a context that carries the current value of the temperature in a certain room. BDA is an extended version of hadoop from Telefonica (called Cosmos). Finally, CEP is an IBM generic enabler to correlate real time events according to programmed rules. The data generated either by CEP or BDA is published on CB. BDA is fed by CB and processed data from CEP. Therefore,

FIWARE allows near real time map/reduce operations over large amounts of data from IoT.

2) COMMUNICATION MODEL

FIWARE Orion context broker supports two pub/sub modes: (i) push method to send information to the broker; (ii) pull to send some information requested by the broker. The push method is for the case that the broker does not queried for the information. In this case, subscribed clients receive the information continuously, every time it is updated. The push method is similar to the publish primitive employed at MQTT message broker.

3) NAMING, NAME RESOLUTION, AND SEMANTICS INTEROPERABILITY

FIWARE relays on DNS naming and name resolution capabilities. Resource names are exposed via NGSI. Ramparany *et al.* [56] have discussed the importance of semantic modeling and semantic-based orchestration in IoT platforms, presenting how FIWARE supports semantic-based components for IoT application development. A street lamps application use case has been implemented, deployed, and evaluated using FIWARE. An *et al.* [59] have provided a solution to interconnect FIWARE to oneM2M platform. A semantic-driven integration approach has been developed to enable static mapping of sensing data between platforms, as well as dynamic semantic interoperability via a semantic proxy. A proof-of-concept has been evaluated in the Santander city, Spain.

4) SERVICES LIFE-CYCLING AND ANALYTICS/AI

FIWARE encompasses a catalog that transparently supports addition of new services as generic enablers. Service life-cycle is supported by OpenStack features. FIWARE encompasses native support for big data and analytics. It also provides business intelligence via Knowage generic enabler. However, at the time of this writing, AI algorithms are not available as architectural components.

5) ENTITY REPRESENTATION

FIWARE device management contains a software representation (called IoT agent) for each backend device connected. NGSI-9/10 interfaces are proposed to gateways, even though other interfaces are supported. Apparently, the MQTT topics employed in many IoT middleware are equivalent to the FIWARE CB pub/sub mechanisms using NGSI-9/10 interface.

6) SECURITY, PRIVACY AND TRUST

FIWARE security depends on NGSI-9/10 interfaces secrecy and authentication. Additionally, FIWARE offers several security enablers: (i) privacy preserving authentication, which issues credentials, presentation policies and verifiers to compute access tokens; (ii) KeyRock identity management to handle several issues related to users access to the platform; (iii) AuthZForce provides an API to get authorization

decisions based on authorization policies as well as authorization requests from policy enforcement points (PEPs). Trust formation is limited to these enablers. Alonso *et al.* [60] have proposed a model to connect FIWARE services authenticated with OAuth 2.0 to electronic identification and trust services (eIDAS) nodes, therefore meeting recent European Union data protection/privacy regulations.

J. IOTIVITY

IoTivity is an open source project hosted by Linux Foundation that implements a reference specification of the open connectivity foundation (OCF). The OCF is a group of industry leaders who is developing a standard specification and certification program to address IoT challenges [61]. One of the main goals of OCF is to provide devices discovery and connectivity. The project is supported by several technology companies like Intel, Qualcomm, LG, Samsung, Cisco, Microsoft, etc. IoTivity provides a software development kit (SDK) in C, C++ and Java languages under Apache 2.0 license. IoTivity is now merging with AllJoyn initiative.

1) ARCHITECTURE AND CLOUD COMPUTING

The IoTivity architecture has two main layers (as illustrated in Figure 3): (i) IoTivity service layer; and (ii) IoTivity base layer. The IoTivity service layer has the sub-layers: (a) Resources encapsulation (RE): it provides common functional modules like broker, builder, container, cache, etc; (b) Services: it contains the service modules that use the functional modules of RE layer. The IoTivity base layer also has two sub-layers: (a) Resource introspection: it is responsible for resource type/properties management; (b) Connectivity abstraction: it performs Wi-Fi, Bluetooth, and BLE abstractions with CoAP. IoTivity has been extended to support many cloud-based services, such as: resource life cycle, security with TLS, and OAuth2 over CoAP, MQTT pub/sub brokering, device-cloud keep-alive, among others.

2) COMMUNICATION MODEL

The IoTivity uses RESTful, BLE, and CoAP with JSON as communication model. Also, it can handle dual-stack IPv4 and IPv6. Lee *et al.* [62] have implemented a health care application for blood glucose, body temperature, and oxygen saturation measurements over IoTivity platform. BLE connectivity was employed to connect sensors to an Android OS application. Elsayed *et al.* [63] have proposed a service discovery platform for heterogeneous IoT environments that runs over IoTivity. The approach called campus as a mashups platform for IoT experimentation (CAMPIE) integrates BLE, MQTT, Thread, ZigBee, Z-Wave, Ethernet, Wi-Fi, etc. A resource proxy is implemented to deal with devices heterogeneity.

3) NAMING, NAME RESOLUTION, AND SEMANTICS INTEROPERABILITY

The specification is not clear about naming neither name resolution strategies. Current Internet technologies are employed for these aims, e.g. URI.
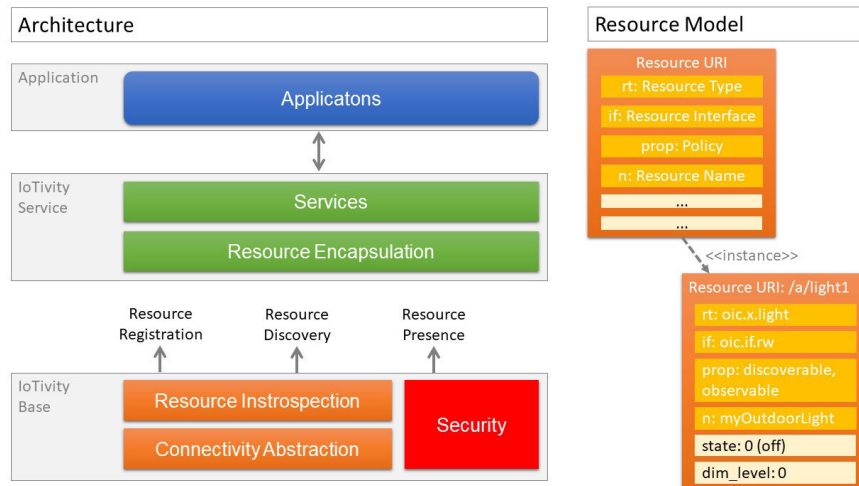
FIGURE 3. IoTivity two layers architecture (Service and Base) and Resource Model (scheme and instance).

4) SERVICES LIFE-CYCLING AND ANALYTICS/AI

Cloud-based device life-cycling is supported via IoTivity stack, which can be instantiated via Docker containers, including Apache Kafka (for data stream processing) and Zookeeper (for hierarchical key-value data storage), MongoDB (database), MQTT message queue, account server, and resource directory. Regarding to AI/ML, no support is provided in the platform.

5) ENTITY REPRESENTATION

As guided by RESTful design, all devices are modeled as resources. The resource URI is composed by: (i) Resource type: identifies the type of resource; (ii) Resource interface: list of interfaces associated to the resource; (iii) Policy: associated with resource life-cycle, i.e. discovery, access, security, etc; and (iv) Resource name: an user-friendly name. Resource life-cycle includes: (a) Resource registration; (b) Resource discovery; (c) Resource presence.

6) SECURITY, PRIVACY AND TRUST

IoTivity uses DTLS/TLS for secure data channel with encryption. There are two main security modules: (i) the security resource manager that takes care of access control; and (ii) the security provisioning manager that is responsible for credentials authentication.

K. LWM2M

The Lightweight Machine to Machine (LwM2M) framework has been created by Open Mobile Alliance (OMA), and has been designed for device management in a machine-to-machine environments [64].

1) ARCHITECTURE AND CLOUD COMPUTING

The LwM2M has a two layers architecture (as illustrated in Figure 4): (i) LwM2M Server: it is typically hosted in private or public data centers and it is responsible to perform higher level requirements like device discovery, communication and security; (ii) LwM2M Client: it is typically hosted in the device and integrated as software library or built-in function of device. The Server can be installed in the cloud, allowing remote handling of Clients.

2) COMMUNICATION MODEL

The LwM2M protocol stack utilizes CoAP over UDP, SMS, TCP, LoRaWAN, and NB-IoT bearers. CoAP is responsible to define the message header, request/response code, message options and transmission mechanisms. LwM2M also defines the UDP binding with CoAP, as mandatory. However, the SMS binding with CoAP is optional. Therefore, the server-client interactions can be performed by both UDP and SMS. Karaagac *et al.* [65] have introduced intermittent connectivity to LwM2M devices. Two new objects have been developed: batch and notify. The batch object enables actions on several devices at once, while the notify object allows devices to send a notification to the server informing they are awake for data retrieval.

3) NAMING, NAME RESOLUTION, AND SEMANTICS INTEROPERABILITY

In order to enable naming resolution, the communication between LwM2M client and LwM2M server follows the URI pattern '/ObjectID/InstanceID /ResourceID', where InstanceID is the name of the object instance on client side. The LwM2M specification keeps open the way how to generate the InstanceID. In others words, it can be a surrogate key or a meaningful name. Silverajan *et al.* [66] have proposed a semantic meta model repository to facilitate data sharing among LwM2M device manufacturers and operators. Karaagac *et al.* [67] have investigated the integration of LwM2M and open platform communications unified architecture (OPC UA), which is a standard for Industry 4.0.
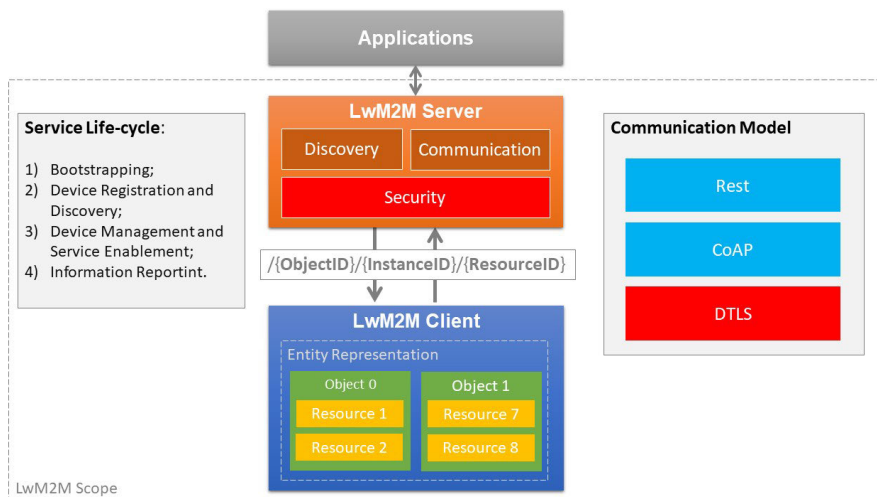
FIGURE 4. LwM2M architecture describing the both server and client side components as well as the service life-cycle and communication model.

4) SERVICES LIFE-CYCLING AND ANALYTICS/AI

The communication between the Server and Client has four logical interfaces: (i) Bootstrapping; (ii) Device Registration and Discovery; (iii) Device Management and Service Enablement; and (iv) Information Reporting. LwM2M offers and online editor in which users can deal with objects and resources life-cycles. Integration to commercial clouds, such as MS Azure, can help on AI/ML/Big data analytics and pattern recognition.

5) ENTITY REPRESENTATION

The LwM2M proposes an entity representation based on two concepts: object and resource. Every LwM2M client has objects and each Object has resources. Every object has a unique predefined identification (ObjectID). In the first release of the LwM2M specifications, it has been defined an initial set of objects for device management purposes. The following list are some of them:

- LwM2M Security (ObjectID = 0): to define security aspects among servers and clients;
- LwM2M Server (ObjectID = 1): to handle data and functions related to management servers;
- Access Control (ObjectID = 2): to handle access rights that servers are allowed to perform over devices on client side.
- Device (ObjectID = 3): to detail information about the device.

In the same way, every resource also has a unique predefined identification called ResourceID. Taking the object 'LwM2M Security' as example, some of its Resources are 'Security Mode' (ResourceID = 2), 'Public Key or Identity' (ResourceID = 3) and 'Server Public Key' (ResourceID = 4).

6) SECURITY, PRIVACY AND TRUST

LwM2M performs secure communications between client and server by using datagram transport layer security (DTLS)

including pre-shared key (PSK) and public key technology to support both kinds of embedded devices (the very limited and more capable ones).

L. MANIOT

ManIoT [68] is a platform for managing of heterogeneous IoT devices and their associated context-aware services. It offers the possibility of running management applications locally, close to managed devices; or globally, in a cloud infrastructure. The local manager supports users' configurations, like turning on or off a lamp. The global/remote manager deals with high level directives for many local domains, e.g. an energy provider policy for those that reduce consumption. Global managers interact with local managers via TCP/IP and take care of high level decisions. Provided services include device discovery, data storage and secure access. ManIoT specifies data and information models to homogenize communication among devices, services and applications. It manages user's access.

1) ARCHITECTURE AND CLOUD COMPUTING

Managers running locally or in the cloud are structured in 5 layers:

- Device layer - It includes support for physical (e.g. RFID reader, SMS modem, smartphone) and virtual devices (google calendar, google talk).
- Communication layer - It supports many standardized communication protocols, like universal plug and play (UPnP), REST and XMPP.
- Adaptation layer - It includes a number of drivers for adapting from ManIoT data model to the supported communication protocols.
- Service layer - It provides several services that support architecture applications.
- Application layer - Offers a web interface for users with visual alerts. Flexibly supports user applications, like home task automation for example.

2) COMMUNICATION MODEL

The proposal supports REST, XMPP, ZigBee and UPnP device communication approaches. Therefore, client/server model is adopted.

3) NAMING, NAME RESOLUTION, AND SEMANTICS INTEROPERABILITY

Devices are named in natural language and have also unique identifiers. Name resolution is provided by a database (MySQL). The relation to DNS is not clear. A context management component provides sensing data contextualization for other services.

4) SERVICES LIFE-CYCLING AND ANALYTICS/AI

A static set of services is offered for: (i) storage, which keeps a history of collected data, events and information regarding devices features; (ii) scheduler, to schedule periodic device readings or configurations; (iii) authentication, for users and devices; (iv) discovery, which enables the discovery of new devices with previously installed drivers; (v) device configuration; (vi) local/global managers communication; (vii) event management, which enables application notification depending on the states of devices; (viii) conflict management, to avoid conflicting configuration of devices; (ix) context management, which offers location- and time-awareness for other services and applications. AI and ML are out of scope.

5) ENTITY REPRESENTATION

There is not a component to represent entities. However, platform services help on interfacing devices and applications. In this sense, context and conflict management are interesting tools.

6) SECURITY, PRIVACY AND TRUST

ManIoT enables user authentication and access control. Applications have limited view to sensor data. Trust formation is not explored.

M. MBDSAS

The management by delegation smart object aware system (MbDSAS) is a hierarchical, distributed approach for IoT gateways and physical devices management [69]. It provides script-based dynamic reconfiguration of physical equipment. Gateways equipped with MbDSAS web services can detect physical devices under their range and prepare a list of connected devices to forward to a top level manager.

1) ARCHITECTURE AND CLOUD COMPUTING

The management by delegation (MbD) approach is based on three level of entities:

- Managed devices (MDes) - They are the heterogeneous devices being managed by the management hierarchy. Devices can export their features, like memory or energy levels, via web services (SOAP or REST).
- Mid-level managers (MLMs) - They typically run embedded at gateways, being responsible to run the

MbDSAS web service (MbDSAS-WS) to configure the gateway and its MDes. Three services are implemented and connected to MbDSAS-WS: (i) MD list builder, which keeps track of gateway connected devices, reporting to TLM; (ii) MbD solutions, like OSGi and Script MIB, to pull scripts from repository and manage their life-cycle at MLM level. These components can run in a local web server, in a regional data center or in the cloud; and (iii) Java, C or TCL run time environments to execute management scripts handled by MbD solutions. Feedback on executed scripts can be cached locally or forwarded to TLMs. The MbDSAS-WS was implemented in PHP and can expose services to be accessed by TLM.

- Top-level managers (TLMs) - It encompasses management applications that: (i) delegate to MLMs specific routines to manage MDes; (ii) control MDes joining and leaving the network; (iii) provide script selection and configuration for MLMs; (iv) exposes web services for users; (v) notify MLMs about new management scripts deployed; (vi) manage scripts repository (script life-cycling). TLMs can relay on cached information and a timer to determine when scripts become out of date. The management application was developed in Java and provides a GUI for human operators. It can be applied to manage experiments, inserting workload, monitoring experiments, collecting performance information and delegating management.

The MbDSAS architecture is aimed at managing configuration and performance of IoT devices, including gateways and sensors/actuators. In other words, IoT adapters could employ MbDSAS TLM to manage gateways and devices of a number of suppliers using REST web services. Architectural components can run in virtual machines at the cloud.

2) COMMUNICATION MODEL

Discovery protocols like link layer discovery protocol (LDP) and universal plug and play protocol (UPnP) are employed at devices level. Web services (SOAP or REST) are combined with IETF script MIB and open service gateway initiative (OSGi), to enable gateway reconfiguration without firmware patching or update. Therefore, the proposal adopts more than one communication model.

3) NAMING, NAME RESOLUTION, AND SEMANTICS INTEROPERABILITY

Despite of current Internet naming, MbDSAS employs a web services name structure to identify services in the platform. The proposal implements script repositories accordingly to service IDs. DNS can be used to reach other domains script repositories and services.

4) SERVICES LIFE-CYCLING AND ANALYTICS/AI

MbDSAS offers a static set of services for IoT management. Novel services need to be manually deployed. AI/ML/Big data software has not been integrated in the proposal.

5) ENTITY REPRESENTATION

Devices are represented by MLM instances. Unfortunately, this representation is limited to network management plane.

6) SECURITY, PRIVACY AND TRUST

Security of script repositories is considered. MbDSAS security relays in the security support of employed technologies (SOAP, XML, REST, LLDP, UPnP, etc).

N. MOBILITYFIRST

MobilityFirst is a clean slate future Internet architecture whose design started in 2010. As the name implies, the project focus is device's mobility. It is grounded on decoupling identifiers from locators (ID/Loc) for entities in the network, including devices, services and contents. Global unique identifiers (GUIDs) can be attributed to all entities. They can be randomly generated or calculated as the output of a mathematical hash functions, i.e. a self-verifying name (SVN). GUIDs are dynamically resolved to network locators, allowing entities to move without losing their identities. A global name resolution service (GNRS) has been proposed to resolve IDs on locators. In last years, MobilityFirst has been increasingly applied to IoT and smart environments.

1) ARCHITECTURE AND CLOUD COMPUTING

MobilityFirst network is composed by content routers (CRs), content producers and consumers, and instances of the GNRS. Content routers support on-path caching of contents by their GUIDs. Therefore, novel data requests can be answered by delivering in-network content copies. In addition, all routing is based on network addresses, e.g. IP addresses. However, GUIDs are employed to update forwarding decisions. Producers that want to share some information ask the GNRS for a GUID and then register a binding between the provided GUID and the network address of the device where the data is located. Consumers inform to their local CR the GUID of a desired content, as well as their own GUIDs to receive the requested contents. CR queries a GNRS instance to resolve the requested GUID to one or more locators. After selecting one of the network addresses suggested by GNRS, CR routes the content request (called GET message) using the traditional host-centric approach. The requested GUID is also inserted in GET message, since CRs update the target network address during the path. In case of producer device mobility, a CR will eventually change the network address as required, a feature called late binding. The architecture also allows the transition to other producers that have the same content. The returning path is independently routed, therefore supporting data consumer mobility.

2) COMMUNICATION MODEL

MobilityFirst adopted a pull communication model in which content consumers indicate the GUID of a desired information. Applying this approach for named content IoT (ICN-IoT) requires gateways to pull data into the sensors.

However, to perform this action gateways need to know the name of the data they wish to obtain. Alternatively, a service-centric networking (SCN) approach for IoT services and devices communication can be built. In SCN-IoT, services talk one another using GUIDs. They also described other adaptations of MobilityFirst for IoT: (i) header compression; (ii) translation between normal MobilityFirst protocol to lightweight version; (iii) mapping of GUID (160 bits) to a lightweight identifier (16 bits).

3) NAMING, NAME RESOLUTION, AND SEMANTICS INTEROPERABILITY

MobilityFirst is grounded in a global name resolution service, which can be distributedly implemented. Semantics interoperability could be built over this GNRS.

4) SERVICES LIFE-CYCLING AND ANALYTICS/AI

An example scenario of SCN-IoT can be composed by two devices: and IoT device (FitBit) and a smartphone; and three services: user step count service, social network and e-health application. The binding between services' GUIDs and their network addresses are stored in GNRS. When the user forgets its FitBit, a novel binding between user step count service and the smartphone address is registered in GNRS. Therefore, the other two applications talk now to the user step count service instance on the smartphone instead of the one in FitBit. This scenario illustrates a service migration from FitBit to Smartphone by rebinding the user step count service GUID to a novel network address. No specific support or application has been found for ML/AI/analytics.

5) ENTITY REPRESENTATION

MobilityFirst can support physical devices representation (digital twins) via service life-cycling. However, no references to this subject were found in the literature.

6) SECURITY, PRIVACY AND TRUST

Self-verifying names ensure data integrity. However, they are not obligatory. MobilityFirst for IoT adopts a service-oriented security model. It can secure IoT services communication using digital signatures. Privacy can be supported by attribute-based encryption.

O. NAMED DATA NETWORKING

Named-data networking (NDN) is an information-centric future Internet architecture. Unlike TCP/IP, which employs host-centric addressing and URLs for content access, NDN adopts a named content approach, in which packets carry only the name of the content an application is interested. In other words, content is searched and delivered directly by its name, instead of host locators. The incorporation of NDN in the IoT landscape has gained a lot of attention in recent years.

1) ARCHITECTURE AND CLOUD COMPUTING

NDN architecture replaces the IP protocol in the Internet stack by a named data protocol for content request

and retrieval. In this new protocol, two packet types are employed: interest and data. The interest packet contains the name of the content required by some application. No host or router addresses are included in the interest packet header. The network forwards and routes interest packets using the content names. Every data packet meets a prior demand informed via the interest packet. There is a one to one relationship. NDN nodes contain three components: (i) forwarding information base (FIB); (ii) pending interest table (PIT); and (iii) content store (CS). When an interest packet arrives at an NDN node, it checks for existing data in the content store (network cache) using the desired information name. If a match happens, it means a previous request for the same content has already been answered and the content is in the cache. Then, the interest packet is deleted and a data packet is delivered via the same network interface of the content query. In case the content is not in CS, the node verifies the pending interest table. If there is a match on the content name, meaning previous interest packets have already queried for the same content, the interest packet is discarded and the network interface is added to the PIT. If there is not a match, a query in the FIB is performed to determine the interfaces for which the desired content can be found on other domains, nodes or content publishers. In this case, a PIT entry is added to register the returning path of the data packets. Regarding cloud-based operation, NDN can run in cloud and/or edge computing scenarios.

2) COMMUNICATION MODEL

NDN originally adopted a pull communication model, in which content is discovered by interest packets forwarded accordingly to their names. The delivery path is obtained from soft-states registered in the nodes when forwarding interest packets from consumers to producers. Therefore, in a typical IoT scenario, the gateways should periodically issue interest packets for sensor's data. Three strategies have been proposed to extend NDN towards data push operations from sensors to gateways: (i) interest notification; (ii) unsolicited data transmissions; (iii) virtual interest polling.

3) NAMING, NAME RESOLUTION, AND SEMANTICS INTEROPERABILITY

NDN names are hierarchically structured, e.g. */inatel.br/building3/room20/en/temperature/sensor/*1. They look similar to URLs. However, they could include self-verifying names generated by hash functions. This naming structure allows name-based aggregation on NDN nodes, i.e. part of the name can be read as a name prefix, allowing access to multiple contents, simultaneously. A name resolution system similar to Internet's DNS has been proposed. Data interoperability can take advantage of NDN naming, which could include encoding and versioning information directly in the content names.

4) SERVICES LIFE-CYCLING AND ANALYTICS/AI

NDN naming structure allows creating namespaces for service registration and discovery. Services can register themselves directly in a named service bus via a local NDN instance using specific name prefixes. Possible peer services can discover registered ones using the same name conventions. Another possible approach is the publication of service's profile as content through an special API. Profiles can include service IDs, access control policies and API to reach the services directly. Semantic matching engines have also been developed to check fitness of available services to emerging demands. Even proposals to extend the semantics of NDN protocols for service discovery and provisioning have already been implemented, e.g. NDN at the edge. It improved NDN packets to identify cloud services by their names.

A machine learning-based mobility management scheme for NDN is available in literature. Significant reduction in handover delay has been reported for mobile IoT nodes. Machine learning is also adopted to optimize interest packets forwarding. NDN FIB compression using artificial neural networks is also a possibility. Deep learning can also optimize NDN forwarding. An approach is to employ content names, selected node interfaces, and interest packet pending states for training deep learning algorithms.

5) ENTITY REPRESENTATION

The representation of things is a service that can be developed for NDN employing service life-cycling extensions as previously described. However, no references to this subject were found in the literature. Therefore, this issue is a research opportunity.

6) SECURITY, PRIVACY AND TRUST

Contents are ciphered by publishers using public key cryptography. Public key certificates can be obtained by certifying authorities in the NDN network. NDN provides name-data integrity and authenticity since the names are the only thing required to receive contents. Despite hierarchical components, content names can have a digital digest of the content itself, allowing integrity check at every node. However, self-verifying names are not obligatory. Also, there is an important limitation in this pull communication model: how to generate interest packets for content whose hash is unknown? In addition, there is a lack of peer authentication in NDN.

P. NOVAGENESIS

NovaGenesis (NG) is a service-oriented, contract-based, information-centric, software-defined future Internet architecture [1], [19] funded by Ministry of Science, Technology, Innovations and Communications in Brazil. Its current implementation can be seen as a platform, since it runs at user space in Linux kernel. NovaGenesis implements a distributed system in which any information processing or exchanging is seen as a service. All entities are named and name bindings
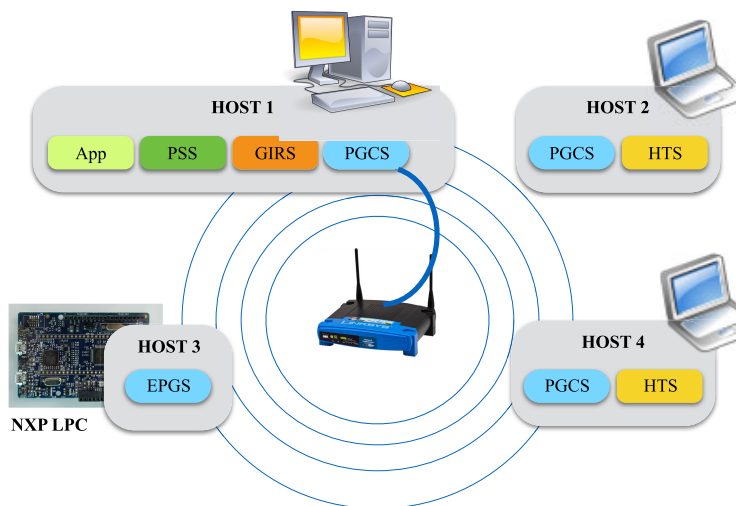
FIGURE 5. NovaGenesis scenario with an EPGS for IoT that is running in Host 3. EPGS is represented by PGCS on Host 1. This PGCS establishes contract with an application (App) interested in IoT samples (e.g. temperature). Just after exposition, discovery and contracting of EPGS, PGCS and App is that data samples are transferred.

are distributedly stored to meet scalability requirements. Also, services (including protocol implementations) organize themselves based on names, name bindings, and contracts to meet semantically rich goals and policies. Every component of the architecture is offered to others by publishing several name-bindings (NBs).

1) ARCHITECTURE AND CLOUD COMPUTING

NovaGenesis software is composed by a set of distributed services, which have several internal objects, called blocks, implementing specific functionalities. Services run at nodes, employing Linux operating system. A domain has a set of nodes, their services and contents. IoT support is based on smart objects, gateways and controllers implemented as services. A smart object represents entities in a network for any purpose. For instance, a Wi-Fi access point can be represented by a smart object service that ''sells'' its forwarding capacities. This smart object can negotiate the Wi-Fi capabilities to potential applications. Another important functionality of NovaGenesis IoT is a gateway, which translates or encapsulates messages from an input protocol to a desired output protocol. Finally, a controller functionality is also implemented as a service, configuring devices accordingly to services need. In the current prototype, these smart object, gateway and controller functionalities are implemented all together in a unique service. NG services can run inside virtual machines and/or Docker containers, locally, in regional data center or at cloud computing facilities.

To support IoT scenarios, NovaGenesis implementation is founded in four core services (Figure 5):

- Publish/subscribe service (PSS): NovaGenesis employs a publish/subscribe model in which NBs (and related contents) are published/subscribed by services through a PSS instance API. This API offers several primitives

to publish, notify, and subscribe names and associated contents (e.g. a .jpg file can be bind to its hash). Therefore, PSS together with the next two core services offers a distributed name resolution service (NRS) with a network cache functionality.

- Generic indirection resolution service (GIRS): Name bindings and associated contents are published/subscribed via PSS, which provides a rendezvous service. However, name bindings or published contents are not stored in the PSS itself. They are stored in a hash table service (HTS). Therefore, GIRS selects the proper HTS to store them.

- Hash table system (HTS): Implements a distributed hash multimap in several nodes. Name bindings can be generated from natural language names (NLNes) or self-verifying names (SVNes). They have a key and one or more values in the format: $<$ key, value(s) $>$. A mathematical hash function is employed for SVNes. A hash code is generated over the entire content or entity unique attributes.

- Proxy/gateway/controller service (PGCS): NovaGenesis requires link layer technologies to deliver its messages. In this context, PGCS provides: (i) encapsulation of messages; (ii) a smart object to represent devices; (iii) bootstrapping functions to initiate communication; (iv) configuration of devices. An embedded version of PGCS was developed and called embedded proxy/gateway service (EPGS).

The integration of PSS, GIRS and HTS is called name resolution and network cache service (NRNCS).

2) COMMUNICATION MODEL

Link layer technologies, e.g. Wi-Fi, LoRa, and IEEE 802.15.4, are employed to deliver NG messages. Therefore,

NG does not use MQTT, CoAP, HTTP, TCP/IP, etc. Sources publish to NG NRNCS anytime they want. Subscribers can asynchronously discover published content or be notified by publishers in case they already know each other. In current prototype, content and NBs should be temporally stored at NRNCS. NRNCS mechanism enables publishers to expose their name bindings (or content) before meeting any subscribers. This enables subscribers to discover possible peers via asynchronous access to previous publications. Based on this discovery mechanism, services can discover, negotiate and contract one another. After contract establishment, publishers can update authorizations to give access to their already published content.

3) NAMING, NAME RESOLUTION, AND SEMANTICS INTEROPERABILITY

To discover existing topics in a domain, a service should query PSS for certain keywords or their hash codes. The published NBs that the service is authorized to see will be returned. Of course, some topics will be private and others not. Contracts are employed to determine the security issues related to access public or private topics. This generic approach allows NG's NRNCS to store and resolve name bindings related to network addresses. In other words, NRNCS allows services to resolve MAC addresses to IP addresses or domain names, covering any required namespace for IoT.

4) SERVICES LIFE-CYCLING AND ANALYTICS/AI

Services life-cycling is contract-based. Contracts are formulated and negotiated after peer services discovery. Contracts set the limitations, responsibilities and clauses of the services provided. The majority of smart environment middleware does not employ contracts right now. Therefore, a possible direction of improvement is to extend them towards state-of-the-art services life-cycling. In this context, smart environment services should be able to: (i) expose their features and capacities to other services; (ii) discover possible peer services in the middleware considering operator policies; (iii) select candidate peer services for contract negotiation; (iv) operation in a contract-based model. An integration of NG services to Spark Big Data tool has been specified in 2016 [36]. An ongoing work is being developed to integrate NG to OpenCog [70] for a smart room scenario.

5) ENTITY REPRESENTATION

A virtual working topology is defined for each application and contains only the components that make sense for it. Through the virtual working topology, an application can read the physical objects state/conditions and/or configure them. PGCS exposes the device features, capabilities and configurations, as well details of the available functions and procedures. In the contrary direction, PGCS reflects to the physical world the configurations required by established contracts.

6) SECURITY, PRIVACY, AND TRUST

NovaGenesis prototype does not implement any security algorithm. However, its security, privacy and trust model was already designed and some of its cornerstones (in addition to traditional techniques) are the name resolution, the support for SVNes, and the formation of trust networks via service contracts.

Q. OPENIOT

OpenIoT [2] is a European FP7 funded project with the aim to provide a middleware platform for semantic interoperability of heterogeneous IoT scenarios integrated to cloud computing. OpenIoT addresses the problem of combining data streams and services from different IoT scenarios. For example, data streams with different units, raw sensor values, etc. It aims at collecting, filtering, contextualizing and selecting data from heterogeneous sensing devices. Data sets are linked using *linked data* concept. It also includes visualization tools for enabling easy development of cloud based IoT applications.

1) ARCHITECTURE AND CLOUD COMPUTING

OpenIoT architecture encompasses seven components: (i) a sensor middleware called extended global sensor network (X-GSN) to collect, filter and aggregate data streams from physical or virtual sensors employing a pub/sub mobile broker; (ii) a cloud database storage called linked stream middleware light (LSM-light) to store data streams and metadata from sensor middleware; (iii) a scheduler to on demand deploy services and give them access to data streams; (iv) a service delivery and utility manager (SD&UM) for service-driven data stream combination and management; (v) a component for specification of service requests (with user interface), which enables scheduling of new services at the scheduler; (vi) a component to select appropriate scripts while visualizing services outputs; and (vii) a component for devices operation. As an example of OpenIoT application, Medvedev *et al.* [71] have developed an architecture to report occurrences in smart city streets. Location, timestamp, voice memos, and video recordings are sent by smart phones to OpenIoT middleware that runs in a local data center. Reports are stored in a database. Voice recognition and reasoning is applied to deliver reports for manual processing when required.

2) COMMUNICATION MODEL

OpenIoT relays in a publish/subscribe middleware called cloud-based publish/subscribe middleware for IoT (CUPUS). CUPUS has two components: (i) a mobile broker that runs on mobile devices to collect data from sensors; (ii) a cloud data processing service. CUPUS offers several subscription mechanisms, which avoid transferring of irrelevant data to the cloud. CUPUS offers elastic pub/sub processing at the cloud.

3) NAMING, NAME RESOLUTION, AND SEMANTICS INTEROPERABILITY

OpenIoT enhances existing ontologies to deal with semantic annotation of measurement units, raw sensor data and points of interest for them. The LSM software implements architecture's ontology, transforming data from virtual sensors (smart objects) into linked data, which are stored in resource description format (RDF). These semantic annotated data are queried using SPARQL. Sensor readings are accessible by several APIs, which can be extended to deal with new sensor types. Naming and name resolution follow current Internet technologies.

4) SERVICES LIFE-CYCLING AND ANALYTICS/AI

Soldatos *et al.* [2] do not comment on dynamic composition of services. Apparently, platform services are fixed and manually introduced. The platform does not include components for AI/ML/Big data.

5) ENTITY REPRESENTATION

OpenIoT supports W3C semantic sensor networks (SSNs) standard for representing physical and virtual world sensors. A sensor representation combines the vision of its measures, features, functionalities and how it processes data. The update of virtual sensors (smart objects) is also based on *wrappers*, such as serial communication, UDP, HTTP, etc.

6) SECURITY, PRIVACY AND TRUST

OpenIoT provides a privacy and security module for user management, authentication and authorization. Authenticated users are represented by *token objects* with expiration time. The token is forwarded from one service to another according to the service chain. The implementation is done using OAuth2.0. Per service authorization is implemented using Apache Shiro.

R. SMARTSANTANDER

It provides a European testbed infrastructure for scientific research and experimentation in the context of a smart city [72]. Besides smart city utilities, like parking, public street lighting, and environmental monitoring, the project provides support for testbed observation and management.

1) ARCHITECTURE AND CLOUD COMPUTING

The project encompasses three tiers: devices, gateways and service platform. Services are provided to support the entire experimentation cycle, including scenario specification (resource selection, configuration, provisioning of hardware images), setup (resource reservation, scheduling, and deployment), and execution (experiment execution, monitoring, data collection, and logging). Services can run virtualized in cloud infrastructure. Devices and gateways employ IEEE 802.15.4, general packet radio service (GPRS), universal mobile telecommunications system (UMTS), digimesh and Wi-Fi. Experimentation with services is provided via

RESTful APIs, JavaScript object notation (JSON) and intelligence data advanced solution (IDAS) from Telefônica I + D. More recently, Sotres *et al.* [73] have discussed lessons learned in terms of interoperability, management, energy harvesting, co-existence with other networks, threads from physical environment (humidity, etc.), and data life-cycling.

2) COMMUNICATION MODEL

In the service layer, a REST API is provided for querying and retrieval of IoT resources accordingly to users need. Meanwhile, the communication among services and gateways is performed via an event bus implemented over the ActiveMQ [74] pub/sub asynchronous model.

3) NAMING, NAME RESOLUTION, AND SEMANTICS INTEROPERABILITY

Naming has an important role in architecture. ActiveMQ topics are employed to support device management. Event types are implemented using google protocol buffers [75]. In addition, the proposal offers a resource registration interface implemented via REST uniform resource identifiers (URIs). To every registered resource a URI is provided. Resources are described using XML documents that contain many node information, including its MAC addresses. A uniform resource name (URN) is employed to uniquely identify resources for experiments [72]. Resource lookup is granted based on resources characteristics using REST GET method. Therefore, SmartSantander name resolution is capable to resolve resources URIs on devices MAC addresses.

4) SERVICES LIFE-CYCLING AND ANALYTICS/AI

Services are statically orchestrated to support architecture functionalities. Sánchez *et al.* [72] does not comment on dynamic service composition at the service layer, i.e. the authors do not discuss proposal's extensibility. An experimentation support subsystem provides measured data analysis. AI/ML/Big data are not a component of the architecture.

5) ENTITY REPRESENTATION

SmartSantander provides an IoT resource manager that represents devices at service level, offering devices registration, monitoring and configuring actions. A testbed runtime configuration component specifies devices available for experimentation, reflecting users intent to devices via resource management. Paganelli *et al.* [76] have developed a framework to run integrated to SmartSantander platform. The aim has been to make things available as RESTful web resources. Digital representatives (smart things) have been developed and tested in the field.

6) SECURITY, PRIVACY AND TRUST

Users are authenticated and can securely select resources (if authorized) for their experiments by employing a reservation system (RS) API. The reservation process returns a private URI for reserved devices.

S. SENSEI

SENSEI [77], [78] is aimed at delivering a managed environment for the interaction of IoT service providers and consumers. The idea was to integrate islands of heterogeneous wireless sensor and actuator networks (WSANs) to future Internet. The proposed architecture relied on RESTful interfaces to create a market of IoT devices and services. Some of the main characteristic of SENSEI proposal are: (i) search, discovery, dynamic composition, and instantiation/deployment of IoT services and devices; (ii) "semantically rich" models and descriptors of sensors, actuators, and processing elements; (iii) device users interact directly with things or resource directories; and (iv) "semantic brokers", execution managers and dynamic resource "creators" that provide more complex interfaces, where users can execute short/long term queries, deploy new devices, and perform more sophisticated queries using meaning (e.g. temperature in room X). Experimental results have been provided for 6LowPAN and ZigBee [78], including a web based access to the architectural components.

1) ARCHITECTURE AND CLOUD COMPUTING

SENSEI architecture is composed by three layers: (i) communication service layer; (ii) real world resources layer; and (iii) application layer. The communication service layer provides connectivity to physical world devices. The real world resource layer provides a unifying abstraction for heterogeneous devices, homogenizing access, managing communities of devices and supporting devices discovery. Finally, the application layer encompasses functionalities for context-aware control and management of resources. Since it is an older project, there is no reference to cloud computing. However, many of the services provided can possibly run as virtual machines in a cloud infrastructure.

2) COMMUNICATION MODEL

Each component of the architecture is accessible by a REST API, as a resource on the web. Therefore, SENSEI communication model is client/server. Sensors are either directly connected to components or accessed via a REST gateway, e.g. when ZigBee is employed.

3) NAMING, NAME RESOLUTION, AND SEMANTICS INTEROPERABILITY

Resource layer offers support for identity management of all entities. It implements an entity dictionary, with an ontology and a mapping table to relate real world entities to their logical representation. An entity lookup service is provided for applications in order to enable semantic rich queries.

4) SERVICES LIFE-CYCLING AND ANALYTICS/AI

Although SENSEI provides a complete life-cycle for wireless sensor and actuator networks, the introduction of new services in architecture is static. For instance, if some use case requires a novel service, let's say a data interoperability tool,

current components are unable to discover this new service and take advantage of it. There is no mention to AI/ML/Big data in references [77], [78].

5) ENTITY REPRESENTATION

SENSEI provides support for representing physical entities as resources that can be exposed to applications. Dynamic resource creation, entity directory and semantic query components allow flexible resource utilization by applications.

6) SECURITY, PRIVACY AND TRUST

Tsiatsis *et al.* [78] does not mention about SPT support on SENSEI. Even though, based on architecture description security is dependent on REST practices.

T. THINGWORX

It is a commercial, cloud-based, event-driven, and object-oriented IoT platform. Abstract models can be defined for things, properties, devices, services, mashups, events, users, data, value streams, data tables, data tags, networks, organizations, and authenticators. Models instances are accessed via RESTful APIs. A list of things in a model can be obtained by accessing https:<Server>/Thingworx/Things. Properties, services, and other features of a thing can be accessed by: https:<Server>/Thingworx/Things/<thingName>. This thing-centric approach is adopted to model entire solutions. Model's inheritance and polymorphism are employed to reduce time to deploy.

1) ARCHITECTURE AND CLOUD COMPUTING

The architecture is organized in three tiers: (i) physical entities models; (ii) platform; (iii) databases. The physical tier contains things, devices, and agents that connect to ThingWorx platform. It also contains the users of the platform that access its resources via web. The platform tier contains core components of the architecture, such as: connectivity server to connect things; foundation server to run IoT services; system monitor; and load balancing. The last tier contains support for several databases, including PostgreSQL, Microsoft SQL, SAP HANA and DataStax Enterprise. Software components of the platform can be hosted on Amazon EC2 cloud.

2) COMMUNICATION MODEL

Accordingly to platform documentation, the following connectivity options are available to devices: (i) a REST API to support HTTP POST from device to the platform; (ii) a proprietary/native AlwaysOn protocol; (iii) proxy-based connection using AlwaysOn protocol for devices that cannot support TLS; (iv) several industrial network adapters; (v) device cloud adapters for Amazon AWS and Microsoft Azure; and (vi) several IoT protocol adapters, including MQTT, CoAP, SigFox, etc. As an example, Jaurkar *et al.* [79] have developed a car parking system based on ThingWorx platform. A perception layer have been deployed to transfer park vacancy data to the cloud using HTTP or FTP. The link layer supports 3G, GSM, Ethernet, Wi-Fi, Bluetooth,

and ZigBee. An application running in a smartphone has been developed for users interaction. Another interesting example of connectivity is provided by Yasmin *et al.* [80] In this work, LoRa sensors are connected using MQTT to a ThingWorx platform in cloud via a Nokia's 5G setup.

3) NAMING, NAME RESOLUTION, AND SEMANTICS INTEROPERABILITY

Unique natural language naming is supported for all entities. Name resolution is achieved by directly searching in abstract model instances. Systems interoperability is provided by connectors. Data tags allow users to define a vocabulary (ontology) for objects and data itself, facilitating their filtering and discovering. Data shapes are employed to structure the data.

4) SERVICES LIFE-CYCLING AND ANALYTICS/AI

Thing models can have associated services which are implemented using server side SQL or JavaScript codes. Services are invoked by a REST API. Services input and output data are formatted accordingly to supported data types. An implementation editor allows users to manually implement their things' services. User application development follows a model-based approach using a GUI. Application development consists in: designing a data model; defining application functional behavior in terms of model abstractions, properties and events; integration of business logic; and connection to third party systems. Dynamic composition is obtained by integrating models in an application. The platform encompasses a big data analytics component, which includes supervised machine learning.

5) ENTITY REPRESENTATION

A thing model is an abstract representation of a device, product, system, people or a logic in business process. Things can have properties, services, events, and subscriptions to changes occurred in other things. Thing model templates facilitate new models development via inheritance of previous thing abstractions. Remote things are used to represent devices outside the ThingWorx platform.

6) SECURITY, PRIVACY AND TRUST

Permissions for users or groups of users are granted at design time and/or run time. Run time permissions control access to things, collections (group of things), and templates. Connectivity security depends on the protocols employed, i.e. RESTful APIs, MQTT, CoAP. Support for encryption, authentication, certificates, and TLS is mentioned in platform documentation.

VI. QUALITATIVE COMPARISON

In this section, it is provided a comparison among the analyzed smart environment proposals considering the aspects selected in Section IV, namely: (i) architecture and cloud computing support; (ii) communication model; (iii) naming, naming resolution and semantic interoperability; (iv) services life-cycling and artificial intelligence; (v) entities

representation; and (vi) security, privacy and trust. The same aspects presented before are discussed and compared. Tables 1 and 2 provide a summary of studied proposals.

Regarding proposed architectures, the general case is to cover from devices up to applications, although some proposals do not include support for physical sensors or actuators, e.g. Arrowhead. Some architectures are standardized, such as ALMANAC (IoT-A) and LwM2M. However, the majority is not. Almost all the proposals adopt TCP/IP stack. While NDN, MobilityFirst and NovaGenesis are clean slate proposals, LwM2M allows for non-TCP/IP protocols in its stack. Cloud-based architectures are a reality, since many of them allow instantiation in data center, e.g. ALMANAC, Aneka, Arrowhead, ClouT, COMPaaS, FIWARE, DIAT, etc. Virtualization is employed to allow scalability and elasticity of services, mainly employing virtual machines. Container-based virtualization is not explored, being an open demand. Also, many platforms are not ready for edge computing, a trend in 5G mobile networks. DIAT, Arrowhead, FIWARE, NDN, MobilityFirst and NovaGenesis adopt a distributed model, in which services can run on edge devices, as well as in regional clouds. The majority of the proposals runs on public clouds. In general, layers cover devices, connectivity, adaptation, virtualization, services, and applications.

Heterogeneity is also present at the communication models. The most common approach is REST, which is an HTTP client/server solution. Publish/subscribe is also common, since MQTT is one of the preferred protocols. Some proposals employ both solutions, such as FIWARE, ALMANAC, Aneka, SmartSantander, AWS, AllJoyn, ThingWorx, etc. Pub/sub is being adopted due to its asynchronous nature and topic (name) based operation. Devices can publish data directly to a broker using named topics, which facilitates data distribution (M2M communication). The most common protocols for device connectivity in the studied platforms are: MQTT, HTTP, AMQP, CoAP, XMPP, 6LowPAN, LwM2M, IEEE 802.15.4, ZigBee, and LoRaWAN. MQTT is supported by ALMANAC, AllJoyn, Aneka, Arrowhead, AWS, FIWARE, ThingWorx. NDN and MobilityFirst rely on push/pull model due to its interest-based communication model. NovaGenesis adopts a pub/sub model, but other approaches can be implemented as well.

Regarding naming and name resolution, the majority of proposals totally relays on DNS and MQTT support. Some additional naming and name resolution services have been developed in many proposals to support specific demands. For instance, ClouT employs a CoAP resource directory, while COMPaaS provides a device's naming structure. LwM2M provides a URL name structure to access devices via their *ResourceIDs*. MbDSAS and ManIoT also provide support for devices IDs. These specific services address punctual demands at each proposal. The most common demand is to relate a device ID to its address. However, many other demands are not addressed. For example, the relationship among information objects (sensor samples) and the host that provides such a measurement. As already demonstrated by

TABLE 1. Comparison of proposals features under the future Internet research point of view.

Proposal	Architecture & Cloud computing	Comm. Model	Naming and Name Resolution	Services Life-cycling & AI/ML/Big Data	Entities Representation	Security, Privacy, and Trust (SPT)
AllJoyn	Decentralized with four layers: transport, connectivity, service and application. Cloud computing resources accessed via gateway and device bridge.	Very diverse, including client/server and pub/sub.	Extends D-Bus natural language naming for several entities.	Name-based service and application discovery and composition. AI/ML/Big data not integrated in the architecture.	Active proxy objects represent distant resources locally.	Comprehensive security. Credentials exchange, certificates, trust anchors.
ALMANAC	IoT-A standard. Layers: resource adapt., data mngt., virtualization, API. Native cloud-based operation.	Pub/sub (MQTT) and client/server (HTTP).	DNS for component name resolution. Semantic framework.	Dynamic compose-ability and federation of platform instances. AI/ML/Big data not integrated in the platform.	Via virtualization layer. Support for complex application requests.	Federation identity and access manager for security and trust.
Aneka	Covers from devices up to applications, including multiple cloud providers.	Adopts cloud provider models, e.g. MQTT, HTTP, AMQP.	Not addressed by proposal. Provides semantic data interoperability.	Service life-cycling depends on the cloud providers support. MapReduce analytics supported. No mention to AI/ML.	Delegated to other platforms.	Aneka assumes current IaaS security is sufficient.
Arrowhead	Covers service orchestration and cloud interop. Limited to software layer. Local or online clouds.	Diverse models: REST, CoAP, XMPP and OPC-UA.	Registration and services interop. Naming and resolution are not clear.	Supports SOA principles , i.e. dynamic compose-ability. Does not include native tools for AI/ML/Big data.	Does not covered.	Authentication and authorization. Trust formation for cross domain clouds.
AWS	MQTT and RESTful HTTP over TCP/IP with support for heterogeneous IoT. Native cloud support.	Client/server and pub/sub	Natural language for many entities	Dynamic composition using lambda functions. Integrated AI/ML/Big data.	Passive JSON-based device shadow	Comprehensive security with support for X.509 certificates
ClouT	Three layers: city infra as service, city platform as a service and city application as a service.	Device wrappers for XMPP, GPRS, CoAP, 6LowPAN and OMA LWM2M. Web services.	DNS and CoAP resource directory for device discovery. Data interoperability.	Employs web services life-cycling. Dynamic compose-able. Data and complex event processing.	IoT device management. Virtual machines for IoT devices, abstracting each device and OS.	Security mechanisms for all layers.
COMPaaS	Three layers: physical devices, event processing and integration, services and applications. Ready to run virtualized.	RESTful, Sub/Notify with RESTful and SOAP.	Device naming. Name resolution not covered. Devices profiles for interoperability.	Static. AI/ML/Big data not integrated.	Logical resources represent physical devices.	N/A.
DIAT	Three layers: virtual object, composite virtual object and service layer. Virtual objects can run locally or in cloud.	Agnostic regarding comm. model. Supports pub/sub.	Based on current Internet technologies.	Supports autonomic service life-cycling. Does not include native support for AI/ML/Big data.	Represents physical and virtual entities.	Employs context-based policies to handle entities access control.
FIWARE	Covers from physical devices up to high end applications. Cloud-based.	Publish/subscribe (similar to MQTT) and client/server	Relays in DNS and NGSI naming.	Dynamic composition of generic enablers. Native support for Big data analytics and business intelligence. AI/ML are not integrated.	Represents physical devices via IoT agents.	Provides security service enablers as well as secure interfaces.
IoTivity	Connectivity, resource introspection and encapsulation, services. Cloud-based services, including Docker containers.	RESTful: CoAP with JSON.	URI composed by Resource Type, Interface, Policy, and Name.	Resource Registration, Discovery and Presence. Data stream processing, key-value storage/querying, AI/ML not integrated.	N/A	DTLS/TLS

Ghodsi *et al.* [18], naming and name resolution are at the core of security, privacy and trust support. How to guarantee provenance if data samples are coming from nodes with volatile identifiers? How to form trust networks among services that do not have unique identifiers. NovaGenesis, NDN and MobilityFirst provide novel naming and name resolution services (GNRS, NDNS and NRNCS) that support not only natural language names, but also self-verifying names (SVNes). They address many of these issues by redesigning naming structures for global reachability [19].

Service life-cycling is another important topic in smart environment, since new services should transparently work together with previous ones, preferably in an autonomous way. Despite the obvious importance, only DIAT provides an autonomous manager for smart environment. NDN services can take advantage of interest packets to consume other service profiles and establish agreements. Such features are community provided extensions to the original design. MobilityFirst encompasses a rich name-based service-life cycling support. NovaGenesis provides service

TABLE 2. Comparison of proposals features under the future Internet research point of view (continuation).

Proposal	Architecture & Cloud computing	Comm. Model	Naming and Name Resolution	Services Life-cycling & AI/ML/Big Data	Entities Representation	Security, Privacy, and Trust (SPT)
LwM2M	Two layers: Lw2M2M server and client.	Restful: CoAP over UDP , TCP, LoRaWAN, NB-IoT, and SMS bearers. Server installed in the cloud. Integration to commercial clouds.	Uses the URI pattern /ObjectID /InstanceID /ResourceID	Bootstrapping, device discovery, mngt. and enablement. AI/ML/Big data via Microsoft Azure.	Uses the well-defined Object-Resource data model	DTLS including Pre-shared key (PSK) and public key technology.
ManIoT	Five layers: device, comm., adaptation, service and application. Local or cloud.	REST, XMPP. ZigBee and UPnP.	Natural language names. Name resolution via MySQL database.	Static. AI/ML/Big data not integrated.	Does not provide resource representation.	Tools for user authentication and access control.
MbDSAS	Addresses management plane for device, mid-level and top-level. Local or cloud.	Client/server.	Employs a proper name structure to identify services.	Static. AI/ML/Big data not integrated.	Representation limited to mngt. plane.	Security of script repository.
NDN	Clean slate architecture with name-based routing and in-network caching.	Pull and push operations.	Hierarchical naming (with support for SVNes). NDNS as a name resolution system similar to DNS.	Extensions provide published service profiles and name-based service life-cycling.	Does not provided in original design. Can be extended to support.	Name-based security. Asymmetric cryptography. Authentication of peers is missing.
MobilityFirst	Clean slate architecture with look-up based-routing and in-network caching	Pull communication and service-based content exchange.	Global unique identifiers for all entities and global name resolution service to map among IDs and locators.	Service life-cycling support using unique IDs. No specific support or application has been found.	No references to digital twins were found.	Self-verifying names for data integrity. Service-oriented security model with digital signatures. Attribute-based encryption.
NovaGenesis	Unlimited layering. Local, regional, or cloud (VM or Docker containers).	Pub/sub. Messages encapsulated over Wi-Fi, ZigBee, BLE, TCP/IP, UDP/IP, and LoRa.	Unlimited naming and hierarchical name resolution.	Dynamic composition via contracts. Integration to Spark Big data specified. Integration to OpenCog artificial general intelligence is on its way.	Service-based physical and logical resource representatives.	Name-based. Contract-based trust network formation. Traditional security.
OpenIoT	One service layer. Cloud-based.	Cloud-based pub/sub middleware for IoT (CUPUS).	Standard naming and name resolution. Semantic annotation and querying.	Static. AI/ML/Big data not integrated.	Standardized representation based on wrappers.	User management, authentication and authorization.
Smart Santander	Three tiers: devices, gateways and services. Services can run virtualized in cloud.	Client/server and pub/sub.	ActiveMQ topics employed for devices management. REST resource registration.	Static. Data analysis of experimental data. AI/ML data not integrated. .	Provides IoT resource manager to represent physical devices.	User authentication and access control.
SENSEI	Three layers: communication, real world and application. Probably can run virtualized in cloud.	Client/server.	Entities dictionary with semantic queries capabilities.	Static. AI/ML/Big data not integrated.	Provides physical resources representation.	N/A
Thingworx	Tree tiers: physical, platform, and databases. Services can run in Amazon cloud.	Client/server and pub/sub.	Unique natural language for all entities.	REST-based compose-ability with manual deployment. Integrated Big Data analytics. Supervised machine learning.	Thing models actively represent devices, products, systems, people.	Comprehensive security. Permission-based.

self-organization, which can be seen as one of the autonomic properties desired for smart places. NovaGenesis also provides contract-based orchestration, which is a unique feature among the studied proposals. DIAT, FIWARE, ALMANAC, Arrowhead, ClouT, MobilityFirst and NovaGenesis support dynamic composability of services. In contrast, the remaining proposals support only static (manual) composability, which will require frequent human intervention, increasing operational expenditure (OPEX).

Many approaches supply mechanisms for service discovery, proofing that reduction on human interference is required and is being addressed by services self-organization. Dynamic composability of services means that services can

work together with minimum human interference, i.e. they can discover and start working as a team. New services (computer programs) can establish collaborative work with existing ones. Dynamic composability is supported by AllJoyn, ALMANAC, Arrowhead, AWS, ClouT, DIAT, FIWARE, NovaGenesis, MobilityFirst and ThingWorx. A feature not supported by COMPaaS, IoTivity, LwM2M, ManIoT, MdBSAS, OpenIoT, SmartSantander, and SENSEI.

The integrated support for artificial intelligence, machine learning, and big data is highly desirable for any IoT platform/solution. However, among the proposals studied, only Amazon AWS and LwM2M (via Microsoft Azure) provide such features. ClouT, FIWARE, and ThingWorx encompass

support for Big data. Possibly, some of the platforms can support IA/ML/Big data tools by adding novel components or APIs to exiting software. However, this is not ready for users. Apparently, commercial proposals are ahead in this matter. Extensions to NDN FIA allow machine learning-based optimizations to content forwarding and caching.

The concept of smart objects or digital twins is also commonly adopted in the investigated proposals. Examples are ALMANAC, DIAT, ClouT, COMPaaS, FIWARE, NovaGenesis, OpenIoT, SmartSantander and SENSEI. This demonstrates that smart objects are a *de facto* solution for the devices interoperability problem. In other words, heterogeneous devices are represented by software objects that interoperate one another in the name of physical entities. ClouT instantiates a virtual machine in the cloud to represent one or more physical devices. NovaGenesis is the unique approach were devices representatives establish contract in the name of things. NDN and MobilityFirst do not offer such feature.

Regarding SPT, all initiatives provide some support. The basic one is user authentication and access authorization to resources. Trust network formation, which means to establish a trustable network of services and resources is supported by some approaches, such as: ALMANAC, Arrowhead, DIAT, NDN and NovaGenesis. Only DIAT employs autonomic security. In general, SPT support in studied proposals could be improved to increase protection, role-based access control and autonomic operation. It is quite concerning that some approaches offer very simplistic solution to a complex problem, leaving important issues without coverage, e.g. traceability, provenance, data privacy and access control.

Tables 1 and 2 also facilitate on identifying the main strengths and weaknesses of all studied proposals. AllJoyn enables diverse M2M communication, comprehensive security, and digital twins, even though the support for cloud applications was recently added. ALMANAC offers dynamic service composability, entities representation, and limited communication models: only HTTP and MQTT. SPT appears to be the weaker point. Aneka provides strong cloud support, but lacks on naming, name resolution, entities representation, and SPT. Arrowhead strengths are service orchestration, cloud interoperability, and diversity of communication models. Entities representation and naming/name resolution are not supported. AWS is a comprehensive proposal, with sound SPT, integrated AI/ML, Big data, naming, name resolution, and native cloud-based operation. However, entities are represented by passive JSON files. ClouT delivers a multitude of device wrappers, resource directory, and dynamic service composability. Security covers all layers, entity representation is also offered. However, there is a lack of AI/ML/Big data support.

COMPaaS strengths are communication models, name and name resolution, and resources representation. SPT, service life-cycling, and limited communication are attention points. DIAT has impressive autonomic IoT services support, as well as entities representation. However, limited communication and naming/name resolution are concerns. FIWARE main

features are cloud-based operation, dynamic service composition, context brokering, digital representatives, and SPT support. Naming and name resolution is limited to current DNS. IoTivity supports cloud operation (including Docker containers) and DTLS/TLS security. However, IoTivity has no support for digital twins, offering limited communication and static service composition. LwM2M is strong on connectivity, providing access to AI/ML/Big data via Microsoft Azure, and traditional security. Dynamic service composition is missing. ManIoT provides naming and name resolution features. However, it does not support dynamic composability and entities representation. ManIoT provides limited SPT functionalities. MbDSAS includes entity representation and service naming. It has similar limitations to ManIoT. NDN and MobilityFirst are name-based clean slate future Internet architectures. NDN adopts a content name-based routing approach, with in-network caching.

MobilityFirst encompasses a different approach grounded on global name resolution service. Content routers rely on GNRS to map global unique IDs to locators, which are indeed used to forward packets in the network. NovaGenesis main features are naming, name resolution, contract-based dynamic service composition, and entities representation. It has limitations regarding current SPT implementation and IA/ML/Big data. OpenIoT main strengths are cloud-based operation and standardized entities representation via wrappers. Concerns include naming, name resolution, dynamic service composability, and SPT. SmartSantander strengths are IoT services for experimentation, data analysis, and devices representation. Limited communication and static service composition are weak points. SENSEI is similar to SmartSantander, with SPT being a major concern. Finally, ThingWorx has strong abstractions to represent physical world and services, comprehensive support for SPT, and natural language naming. As can be seen, there are important similarities and differences among proposals, but few of them cover most of the requirements considered in this article.

A. OPEN RESEARCH ISSUES

The diversity of technologies employed in smart environments directly influences the key aspects that should be selected for qualitative comparison. In this article, a set of key aspects aligned to future Internet research has been selected, providing novel insights on the limitations of current proposals and previous literature reviews. The opposite is also true, the three future Internet proposals (NDN, Mobility-First, and NovaGenesis) also have unimplemented aspects already available in TCP/IP-based proposals. Even though, some challenges still remain as opportunities for further research:

- Improvement and expansion of the key aspects considered in reviews and architecture designs - This is huge challenge, since the spectrum of key aspects presented in Section III is enormous. A very broad survey on the key aspects and enablers for smart environments is still missing in literature.

- Importance sampling - Many surveys give a lot of attention to aspects less relevant than others, specially regarding evolution towards new generation architectures, such as future Internet or 5th generation mobile networks. Quantitative techniques could be used to determine relative importance of key aspects and enablers, specially regarding new generation networks.
- Synergy among design ingredients - In general, smart environment platforms can better explore synergies among adopted enablers. Tables 1 and 2 give several examples. Naming and name resolution is superficially explored by the large majority of proposals, despite its importance for other architectural aspects, specially security, privacy, and trust [18].
- Service-orientation is another example. Manual intervention is predominant. Approximately half of the platforms employ dynamic service composition. Devices are not represented by services in many proposals, a fundamental feature for IoT and smart environments, since smart objects provide an autonomous bridge between physical and virtual worlds. Security does not take advantage of service level agreements to form trust networks between smart objects and high level applications.
- Other FIA and 5G enablers, such as network programmability, network function virtualization, self-organizing networking, AI, ML, distributed immutable information databases (e.g. blockchain, tangle, etc.), context-awareness, cloud-based radio access network, virtual functions elasticity, in-network computing, among others, are not being cohesively integrated in current smart environments. The majority of proposals in Tables 1 and 2 does not employ any of these enablers.
- Integrated support for AI, ML, and Big data is highly desirable due to its potential to take advantage of available data and determine trends, generating value to processes and business. While IoT is analogous to nervous systems in biology, AI/ML/Big data is similar to the brain. There is no intelligent decision without nervous systems. Possibly, in the majority of solutions a developer can integrate AI/ML/Big data software to existing implementations. However, what is wanted is that such software be already integrated in the platforms, without the need for further implementations. Another related issue concerns the usage of metamorphic computing embedded in devices. None of the studied approaches are integrated to such distributed, hardware-based AI (called neuromorphic computing).
- Another open research opportunity is *things economy*. Monetizing sensors and actuators through micro payments is another open opportunity. Services would contract sensors accordingly to demand, performing micro transactions to monetize their owners. Digital wallets could be used to store cryptoassets received in payment for measures. The same applies for actuators. A data market place could be integrated to IoT platforms or become accessible by well know APIs.

- Novel naming and naming resolution approaches are required to denote IoT devices, services, users, etc. NovaGenesis and MobilityFirst apply self-verifying naming to all entities in a domain. NDN applies SVNes exclusively for content. Ghodsi *et al.* [18] already demonstrated the huge advantages of self-verifying names in architectures. However, none of the protocols in current Internet architecture employs this kind of naming approach. Not only hierarchical addressing or natural language naming is important, but also self-verifying names, which increase security from architecture foundation.
- In NovaGenesis, sensors and actuators have a representative service that "sells" their features to interested applications. Self-organization is adopted to enable name-based discovery and contracting. Contract-based operation is a unique feature of NovaGenesis. However, SPT in IoT platforms requires trustworthy operation. Digital twins should be involved in "selling" what their physical counterparts can do. Contracts can be established and monetized. *Smart contracts* are a promising technology to automate things economy and make it more safe, since computer programs can be run from Blockchain or other equivalent distributed ledger technology. Users would like to be sure that their programs are exactly the ones they published. Deterministic building (or integral computing) is on its infancy — a large opportunity for researchers and future smart environments.

Future research and development should more deeply incorporate FIA and 5G ingredients in IoT/smart environment platforms, improving their performance, security, flexibility, extendability, autonomy, etc. Not only small medium-sized enterprises, but also big companies should dig deeper into what is being done in these areas to take advantage of emerging technologies in their already successful platforms.

VII. CONCLUSION

This paper provides an overview of the *status quo* of proposals for smart environment, giving a taxonomy of approaches, and analyzing their support for a selected set of key future Internet enablers. Solutions have been analyzed in terms of architecture, communication model, naming and name resolution, services and data life-cycling, representation of physical devices, security, privacy, and trust. It has been observed a big diversity of scopes, technologies and support for these requirements. Even though, comparison regarding some important aspects was possible. The conclusion is that heterogeneity came to stay and solutions need to be generic enough to interoperate to a number of alternatives. Also, publish/subscribe communication model appears to have significant advantages, which justify its broad adoption. However, there are alternatives such as push/pull mode used in the NDN and MobilityFirst projects that deserve further investigation. Future Internet proposals clearly show that there

are significant advantages to start security design by naming and name resolution, especially by adopting self-certifying names. It is known that naming and name resolution plays a key role in security, privacy and trust. Even so, most of the proposals are stuck to the limitations of the current Internet model. Future Internet architectures have exciting advantages that can be explored to build the smart environments of the future. In addition, the dynamic composition of smart city services is not supported in many approaches, certainly increasing operational expenditure. The interoperability of devices is a problem solved. Smart objects (digital twins) are being employed to represent heterogeneous nodes, providing a common language for M2M communication. Ingredients from future Internet research, such as SDN/NFV, ICN and SCN are systematically being adopted for smart environments design. Many others are to come. AI, machine learning, and big data support are missing not only in TCP/IP-based approaches, but also in future Internet-based.

REFERENCES

[1] A. M. Alberti, G. D. Scarpioni, V. J. Magalhães, S. A. Cerqueira, J. J. P. C. Rodrigues, and R. da R. Righi, "Advancing Novagenesis architecture towards future internet of Things," *IEEE Internet Things J.*, vol. 6, no. 1, pp. 215–229, Feb. 2019.

[2] J. Soldatos *et al.*, "OpenIoT: Open source Internet-of-Things in the cloud," in *Interoperability and Open-Source Solutions for the Internet of Things* (Lecture Notes in Computer Science), vol. 9001, Ž. I. Podnar, K. Pripužić, and M. Serrano, Eds. Cham, Switzerland: Springer, 2015.

[3] M. Taneja and A. Davy, "Resource aware placement of IoT application modules in Fog-cloud computing paradigm," in *Proc. IFIP/IEEE Symp. Integr. Netw. Service Manage. (IM)*, May 2017, pp. 1222–1228.

[4] J. von Uexküll and H. Girardet. (2005). *Shaping Our Future: Creating The World Future Council.* [Online]. Available: https://digital.library.unt.edu/ark:/67531/metadc13722/

[5] C. Formisano, D. Pavia, L. Gurgen, T. Yonezawa, J. A. Galache, K. Doguchi, and I. Matranga, "The advantages of IoT and cloud applied to smart cities," in *Proc. 3rd Int. Conf. Future Internet of Things Cloud*, Aug. 2015, pp. 325–332.

[6] J. H. Lee, M. G. Hancock, and M.-C. Hu, "Towards an effective framework for building smart cities: Lessons from Seoul and San Francisco," *Technol. Forecasting Social Change*, vol. 89, pp. 80–99, Nov. 2014.

[7] G. Tselentis, J. Domingue, A. Galis, A. Gavras, and D. Hausheer, *Towards the Future Internet - A European Research Perspective*. Amsterdam, The Netherlands: IOS Press, 2009.

[8] G. Tselentis, A. Galis, A. Gavras, S. Krco, V. Lotz, E. Simperl, B. Stiller, and T. Zahariadis, *Emerging Trends from European Research*. Amsterdam, The Netherlands: IOS Press, 2010.

[9] J. Domingue, A. Galis, A. Gavras, T. Zahariadis, D. Lambert, F. Cleary, P. Daras, S. Krco, and H. Müller, *The Future Internet Future Internet Assembly 2011: Achievements and Technological Promises*. Berlin, Germany: Springer-Verlag, 2011.

[10] F. Alvarez, F. Cleary, P. Daras, G. A. Domingue, J., A. Garcia, A. Gavras, S. Karnourskos, L. M.-S. Krco, S., V. Lotz, S. E. Müller, H. A.-M. Sassen, H. Schaffers, B. Stiller, G. Tselentis, P. Turkama, and T. Zahariadis, *The Future Internet*. Berlin, Germany: Springer-Verlag, 2012.

[11] A. Galis, Eds., *The Future Internet: Future Internet Assembly 2013: Validated Results and New Horizons*. Berlin, Germany: Springer-Verlag, May 2013.

[12] P. Stuckmann and R. Zimmermann, "European research on future Internet design," *IEEE Wireless Commun.*, vol. 16, no. 5, pp. 14–22, Oct. 2009.

[13] J. M. Hernández-Muñoz, J. B. Vercher, L. Muñoz, J. A. Galache, M. Presser, L. A. H. Gómez, and J. Pettersson, "Smart cities at the forefront of the future Internet," in *The Future Internet* (Lecture Notes in Computer Science), vol. 6656. Berlin, Germany: Springer, 2011, pp. 447–462. [Online]. Available: http://link.springer.com/10.1007/978-3-642-20898-0_32

[14] D. Miorandi, S. Sicari, F. De Pellegrini, and I. Chlamtac, "Internet of things: Vision, applications and research challenges," *Ad Hoc Netw.*, vol. 10, pp. 1497–1516, Sep. 2012.

[15] A. M. Alberti, "A conceptual-driven survey on future Internet requirements, technologies, and challenges," *J. Brazilian Comput. Soc.*, vol. 19, no. 3, pp. 291–311, 2013.

[16] A. M. Alberti, M. M. Bontempo, J. R. D. Santos, A. C. Sodré, and R. Da R. Righi, "NovaGenesis applied to information-centric, service-defined, trustable IoT/WSAN control plane and spectrum management," *Sensors*, vol. 18, no. 9, p. 3160, 2018. [Online]. Available: https://www.mdpi.com/1424-8220/18/9/3160

[17] I. Farris, T. Taleb, Y. Khettab, and J. Song, "A survey on emerging SDN and NFV security mechanisms for IoT systems," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 1, pp. 812–837, 1st Quart., 2019.

[18] A. Ghodsi, T. Koponen, J. Rajahalme, P. Sarolahti, and S. Shenker, "Naming in content-oriented architectures," in *Proc. SIGCOMM Workshop Inf.-Centric Netw.*, New York, NY, USA, 2011, pp. 1–6, doi: 10.1145/2018584.2018586.

[19] A. M. Alberti, M. A. F. Casaroli, D. Singh, and R. R. Righi, "Naming and name resolution in the future Internet: Introducing the NovaGenesis approach," *Future Gener. Comput. Syst.*, vol. 67, pp. 163–179, Feb. 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X16302643

[20] Z. Sheng, S. Yang, Y. Yu, A. Vasilakos, J. McCann, and K. Leung, "A survey on the IETF protocol suite for the Internet of Things: Standards, challenges, and opportunities," *IEEE Wireless Commun.*, vol. 20, no. 6, pp. 91–98, Dec. 2013.

[21] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of Things for smart cities," *IEEE Internet Things J.*, vol. 1, no. 1, pp. 22–32, Feb. 2014.

[22] G. Fortino, A. Guerrieri, W. Russo, and C. Savaglio, *Internet of Things Based on Smart Objects*, G. Fortino and P. Trunfio, Eds., Cham, Switzerland: Springer, 2014, pp. 1–27. [Online]. Available: http://link.springer.com/content/pdf/10.1007/978-3-319-00491-4.pdf\nhttp://link.springer.com/10.1007/978-3-319-00491-4

[23] M. A. Razzaque, M. Milojevic-Jevric, A. Palade, and S. Clarke, "Middleware for Internet of Things: A survey," *IEEE Internet Things J.*, vol. 3, no. 1, pp. 70–95, Feb. 2016.

[24] M. A. A. da Cruz, J. J. P. C. Rodrigues, J. Al-Muhtadi, V. V. Korotaev, and V. H. C. de Albuquerque, "A reference model for Internet of Things middleware," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 871–883, Apr. 2018.

[25] J. Guth, U. Breitenbücher, M. Falkenthal, P. Fremantle, O. Kopp, F. Leymann, and L. Reinfurt, *A Detailed Analysis of IoT Platform Architectures: Concepts, Similarities, and Differences*. Singapore: Springer, 2018, pp. 81–101.

[26] M. A. A. da Cruz, J. J. P. C. Rodrigues, A. K. Sangaiah, J. Al-Muhtadi, and V. Korotaev, "Performance evaluation of IoT middleware," *J. Netw. Comput. Appl.*, vol. 109, pp. 53–65, May 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S108480451830064X

[27] P. Sethi and S. R. Sarangi, "Internet of Things: Architectures, protocols, and applications," *J. Elect. Comput. Eng.*, vol. 2017, Jan. 2017, Art. no. 9324035, doi: 10.1155/2017/9324035.

[28] A. H. Ngu, M. Gutierrez, V. Metsis, S. Nepal, and Q. Z. Sheng, "IoT Middleware: A survey on issues and enabling technologies," *IEEE Internet Things J.*, vol. 4, no. 1, pp. 1–20, Feb. 2017.

[29] A. Farahzadi, P. Shams, J. Rezazadeh, and R. Farahbakhsh, "Middleware technologies for cloud of things: A survey," *Digit. Commun. Netw.*, vol. 4, no. 3, pp. 176–188, Aug. 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2352864817301268

[30] E. F. Z. Santana, A. P. Chaves, M. A. Gerosa, F. Kon, and D. S. Milojicic, "Software platforms for smart cities: Concepts, requirements, challenges, and a unified reference architecture," *ACM Comput. Surv.*, vol. 50, no. 6, pp. 78:1–78:37, Nov. 2018, doi: 10.1145/3124391.

[31] H. Hejazi, H. Rajab, T. Cinkler, and L. Lengyel, "Survey of platforms for massive IoT," in *Proc. IEEE Int. Conf. Future IoT Technol. (Future IoT)*, Jan. 2018, pp. 1–8.

[32] M. Nitti, V. Pilloni, G. Colistra, and L. Atzori, "The virtual object as a major element of the Internet of Things: A survey," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 2, pp. 1228–1240, 2nd Quart., 2015.

[33] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, "Context aware computing for the Internet of Things: A survey," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 414–454, 1st Quart., 2014.

[34] O. B. Sezer, E. Dogdu, and A. M. Ozbayoglu, "Context-aware computing, learning, and big data in Internet of Things: A survey," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 1–27, Feb. 2018.

[35] V. Gazis, "A survey of standards for machine-to-machine and the Internet of Things," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 1, pp. 482–511, 1st Quart., 2016.

[36] A. M. Alberti, E. S. dos Reis, R. da R. Righi, V. M. Muñoz, and V. Chang. (Feb. 2016). *Converging Future Internet, Things, and Big Data: An Specification Following Novagenesis Model*. [Online]. Available: https://eprints.soton.ac.uk/387089/

[37] O. Tomanek and L. Kencl, "Security and privacy of using AllJoyn IoT framework at home and beyond," in *Proc. 2nd Int. Conf. Intell. Green Building Smart Grid (IGBSG)*, Jun. 2016, pp. 1–6.

[38] P. Masek, R. Fujdiak, K. Zeman, J. Hosek, and A. Muthanna, "Remote networking technology for IoT: Cloud-based access for AllJoyn-enabled devices," in *Proc. 18th Conf. Open Innov. Assoc. Seminar Inf. Secur. Protection Inf. Technol. (FRUCT-ISPIT)*, Apr. 2016, pp. 200–205.

[39] M. Villari, A. Celesti, M. Fazio, and A. Puliafito, "AllJoyn Lambda: An architecture for the management of smart environments in IoT," in *Proc. Int. Conf. Smart Comput. Workshops*, Nov. 2014, pp. 9–14.

[40] D. Bonino, M. T. D. Alizo, A. Alapetite, T. Gilbert, M. Axling, H. Udsen, J. A. C. Soto, and M. Spirito, "ALMANAC: Internet of Things for smart cities," in *Proc. 3rd Int. Conf. Future Internet Things Cloud*, Aug. 2015, pp. 309–316. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7300833

[41] S. Meyer, A. Ruppen, and C. Magerkurth, "Internet of Things-aware process modeling: Integrating IoT devices as business process resources," in *Proc. Int. Conf. Adv. Inf. Syst. Eng.* Berlin, Germany: Springer, 2013, pp. 84–98.

[42] D. Bonino, M. T. D. Alizo, C. Pastrone, and M. Spirito, "WasteApp: Smarter waste recycling for smart citizens," in *Proc. Int. Multidisciplinary Conf. Comput. Energy Sci. (SpliTech)*, Jul. 2016, pp. 1–6.

[43] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Generat. Comput. Syst.*, vol. 29, no. 7, pp. 1645–1660, 2013.

[44] D. S. S. Subhash, R. A. Ramchandra, P. T. Nagnath, S. D. Rajbhoj, and A. M. Jagtap, "Dynamic provisioning of resources in cloud computing using Aneka," in *Proc. Int. Conf. I-SMAC IoT Social Mobile, Anal. Cloud (I-SMAC)*, Feb. 2017, pp. 905–908.

[45] P. Varga and C. Hegedus, "Service interaction through gateways for inter-cloud collaboration within the arrowhead framework," in *Proc. 5th IEEE Wireless VITAE*, Dec. 2015.

[46] J. Jokinen, T. Latvala, and J. L. M. Lastra, "Integrating smart city services using Arrowhead framework," in *Proc. 42nd Annu. Conf. IEEE Ind. Electron. Soc.*, Oct. 2016, pp. 5568–5573.

[47] H. Derhamy, J. Eliasson, J. Delsing, and P. Priller, "A survey of commercial frameworks for the Internet of Things," in *Proc. IEEE 20th Conf. Emerg. Technol. Factory Automat. (ETFA)*, Sep. 2015, pp. 1–8.

[48] D. Kozma, P. Varga, and C. Hegedüs, "Supply chain management and logistics 4.0 - A study on arrowhead framework integration," in *Proc. 8th Int. Conf. Ind. Technol. Manage. (ICITM)*, Mar. 2019, pp. 12–16.

[49] A. Bhatnagar, V. Sharma, and G. Raj, "IoT based car pollution detection using AWS," in *Proc. Int. Conf. Adv. Comput. Commun. Eng. (ICACCE)*, Jun. 2018, pp. 306–311.

[50] W. Tärneberg, V. Chandrasekaran, and M. Humphrey, "Experiences creating a framework for smart traffic control using AWS IOT," in *Proc. 9th Int. Conf. Utility Cloud Comput.*, Dec. 2016, pp. 63–69.

[51] T. Yonezawa, I. Matranga, J. A. Galache, H. Maeomichi, L. Gurgen, and T. Shibuya, "A citizen-centric approach towards global-scale smart city platform," in *Proc. Int. Conf. Recent Adv. Internet of Things (RIoT)*, Apr. 2015, pp. 1–6, doi: 10.1109/RIOT.2015.7104913.

[52] I. Realtime. *Openfire Project*. (Accessed: Apr. 2017). [Online]. Available: https://www.igniterealtime.org/ projects/openfire

[53] L. A. Amaral, R. T. Tiburski, E. de Matos, and F. Hessel, "Cooperative middleware platform as a service for Internet of Things applications," in *Proc. 30th Annu. ACM Symp. Appl. Comput.*, New York, NY, USA, 2015, pp. 488–493, doi: 10.1145/2695664.2695799.

[54] R. da R. Righi, E. S. dos Reis, G. Rostirolla, C. A. da Costa, and A. M. Alberti, "Exploring cloud elasticity on developing an EPCGlobal-compliant middleware," in *Proc. IEEE Int. Conf. RFID (RFID)*, Orlando, FL, USA, May 2016, pp. 43–46, doi: 10.1109/RFID.2016.7488003.

[55] C. Sarkar, A. U. N. S. N., R. V. Prasad, A. Rahim, R. Neisse, and G. Baldini, "DIAT: A scalable distributed architecture for IoT," *IEEE Internet Things J.*, vol. 2, no. 3, pp. 230–239, Jun. 2015. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7000513

[56] F. Ramparany, F. G. Marquez, J. Soriano, and T. Elsaleh, "Handling smart environment devices, data and services at the semantic level with the FI-WARE core platform," in *Proc. IEEE Int. Conf. (Big Data)*, Oct. 2014, pp. 14–20.

[57] P. Fernández, J. M. Santana, S. Ortega, A. Trujillo, J. P. Suárez, C. Domínguez, J. Santana, and A. Sánchez, "SmartPort: A platform for sensor data monitoring in a seaport based on FIWARE," *Sensors*, vol. 16, no. 3, p. 417, Mar. 2016. [Online]. Available: http://www.mdpi.com/1424-8220/16/3/417

[58] R. T. Fielding, "Architectural styles and the design of network-based software architectures," Ph.D. dissertation, Dept. Inf. Comput. Sci., Univ. California, Irvine, CA, USA, 2000. [Online]. Available: https://www.ics.uci.edu/fielding/pubs/dissertation/fielding_dissertation.pdf

[59] J. An, F. Le Gall, J. Kim, J. Yun, J. Hwang, M. Bauer, M. Zhao, and J. Song, "Toward global IoT-enabled smart cities interworking using adaptive semantic adapter," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 5753–5765, Jun. 2019.

[60] Á. Alonso, A. Pozo, J. Choque, G. Bueno, J. Salvachúa, L. Diez, J. Marín, and P. L. C. Alonso, "An identity framework for providing access to FIWARE OAuth 2.0-based services according to the eIDAS European regulation," *IEEE Access*, vol. 7, pp. 88435–88449, 2019.

[61] IoTivity Project. *Iotivity*. Accessed: Nov. 12, 2019. [Online]. Available: https://www.iotivity.org

[62] J.-C. Lee, J.-H. Jeon, and S.-H. Kim, "Design and implementation of healthcare resource model on IoTivity platform," in *Proc. Int. Conf. Inf. Commun. Technol. Converg. (ICTC)*, Oct. 2016, pp. 887–891.

[63] K. Elsayed, M. A. B. Ibrahim, and H. S. Hamza, "Service discovery in heterogeneous IoT environments based on OCF/IoTivity," in *Proc. 15th Int. Wireless Commun. Mobile Comput. Conf. (IWCMC)*, Jun. 2019, pp. 1160–1165.

[64] O. M. Alliance. *Oma*. Accessed: Nov. 12, 2019. [Online]. Available: http://openmobilealliance.org/about-oma/work-program/m2m-enablers/

[65] A. Karaagac, M. VanEeghem, J. Rossev, B. Moons, E. DePoorter, and J. Hoebeke, "Extensions to LwM2M for intermittent connectivity and improved efficiency," in *Proc. IEEE Conf. Standards Commun. Netw. (CSCN)*, Oct. 2018, pp. 1–6.

[66] B. Silverajan, H. Zhao, and A. Kamath, "A semantic meta-model repository for lightweight M2M," in *Proc. IEEE Int. Conf. Commun. Syst. (ICCS)*, Dec. 2018, pp. 468–472.

[67] A. Karaagac, N. Verbeeck, and J. Hoebeke, "The integration of LwM2M and OPC UA: An interoperability approach for industrial IoT," in *Proc. IEEE 5th World Forum Internet of Things (WF-IoT)*, Apr. 2019, pp. 313–318.

[68] J. B. Antunes, I. L. Dénes, M. Santos, T. O. Castro, D. F. Macedo, and A. L. D. Santos, "ManIoT: Uma Plataforma para Gerenciamento de Dispositivos da Internet das coisas," in *Proc. Simpósio Brasileiro de Redes de Computadores Sistemas Distribuidos (SBRC)*, 2016, p. 14. [Online]. Available: http://sbrc2016.ufba.br/downloads/WGRS/ST1-1.pdf

[69] M. A. Marotta, F. J. Carbone, J. J. C. de Santanna, and L. M. R. Tarouco, "Through the Internet of Things—A management by delegation smart object aware system (MbDSAS)," in *Proc. IEEE 37th Annu. Comput. Softw. Appl. Conf.*, Jul. 2013, pp. 732–741.

[70] D. Hart and B. Goertzel, "Opencog: A software framework for integrative artificial general intelligence," in *Proc. 1st AGI Conf. Artif. Gen. Intell.*, Mar. 2008, pp. 468–472. [Online]. Available: http://www.booksonline.iospress.nl/Content/View.aspx?piid=8338

[71] A. Medvedev, A. Zaslavsky, S. Khoruzhnikov, and V. Grudinin, "Reporting road problems in smart cities using OpenIoT framework," in *Interoperability and Open-Source Solutions for the Internet of Things*. I. P. Žarko, K. PripuĂ¿ić, and M. Serrano, Eds., Cham, Switzerland: Springer, 2015, pp. 169–182.

[72] L. Sánchez, V. Gutiérrez, J. A. Galache, P. Sotres, J. R. Santana, J. Casanueva, and L. Muñoz, "SmartSantander: Experimentation and service provision in the smart city," in *Proc. 16th Int. Symp. Wireless Pers. Multimedia Commun. (WPMC)*, Jun. 2013, pp. 1–6.

[73] P. Sotres, J. R. Santana, L. Sánchez , J. Lanza, and L. Muñoz, "Practical lessons from the deployment and management of a smart city Internet-of-Things infrastructure: The smartsantander testbed case," *IEEE Access*, vol. 5, pp. 14309–14322, 2017.

[74] B. Snyder, D. Bosanac, and R. Davies, *ActiveMQ in Action*. Greenwich, CT, USA: Manning Publications Co., 2011.

[75] K. Varda, "Protocol buffers: Google's data interchange format," Google Open Source Blog, Mountain View, CA, USA, Tech. Rep., Jul. 2018. [Online]. Available: http://google-opensource.blogspot.com/2008/07/protocol-buffers-googles-data.html

[76] F. Paganelli, S. Turchi, and D. Giuli, "A Web of things framework for RESTful applications and its experimentation in a smart city," *IEEE Syst. J.*, vol. 10, no. 4, pp. 1412–1423, Dec. 2016.

[77] M. Presser, P. M. Barnaghi, M. Eurich, and C. Villalonga, "The SENSEI project: Integrating the physical world with the digital world of the network of the future," *IEEE Commun. Mag.*, vol. 47, no. 4, pp. 1–4, Apr. 2009.

[78] V. Tsiatsis, A. Gluhak, T. Bauge, F. Montagut, J. Bernat, M. Bauer, C. Villalonga, P. M. Barnaghi, and S. Krco, "The SENSEI real world Internet architecture," in *Towards Future Internet—Emerging Trends From European Research*, G. Tselentis, A. Galis, A. Gavras, S. Krco, V. Lotz, E. P. B. Simperl, B. Stiller, and T. B. Zahariadis, Eds., IOS Press, 2010, pp. 247–256. [Online]. Available: http://dx.doi.org/10.3233/978-1-60750-539-6-247

[79] H. V. Jaurkar, G. N. Mulay, and V. Gohokar, "Parking guidance system using Internet of Things," in *Proc. Int. Conf. Inventive Comput. Technol. (ICICT)*, vol. 1, Aug. 2016, pp. 1–6.

[80] R. Yasmin, J. Petäjäjärvi, K. Mikhaylov, and A. Pouttu, "On the integration of LoRaWAN with the 5G test network," in *Proc. IEEE 28th Annu. Int. Symp. Pers. Indoor, Mobile Radio Commun. (PIMRC)*, Oct. 2017, pp. 1–6.

ANTONIO MARCOS ALBERTI received the M.Sc. and Ph.D. degrees in electrical engineering from Campinas State University (Unicamp), Campinas, Brazil, in 1998 and 2003, respectively. He has been an Associate Professor and a Researcher with the National Institute of Telecommunications (INATEL), Brazil, since 2004. In 2012, he was a Visiting Researcher with the Future Internet Department, ETRI, South Korea. Since 2008, he has been designing and implementing a future Internet architecture called NovaGenesis. Since 2013, he has been acting as a Coordinator of the Information and Communications Technologies (ICT) Laboratory, INATEL. He has authored or coauthored over 100 articles in refereed international journals and conferences.

MATEUS A. S. SANTOS received the Ph.D. degree from the Universidade de São Paulo (USP), in 2014. From 2013 to 2014, he was a Research Scholar with the Inter-Networking Research Group, UC Santa Cruz. He was also a Postdoctoral Researcher with the University of Campinas (UNICAMP), from 2014 to 2016. He is currently with Ericsson Research, Brazil. His research interests include software-defined networking, network functions virtualization, and network automation.

RICARDO SOUZA received the master's degree from the University of Campinas (UNICAMP), in 2011. He is currently a Senior Researcher with Ericsson Research, Brazil. His main research interests include distributed systems, the Internet of Things, mobile and networked robotics, and artificial intelligence.

HIRLEY DAYAN LOURENÇO DA SILVA received the B.S.E.E. degree with an emphasis in telecommunications from the National Institute of Telecommunications (INATEL), in Brazil, in 2000. He is with Ericsson Research, Development, & Innovation Facility, in Brazil, where he joined, in 2000. He was an Architect in many smart cities and smart industries related projects. His current activity includes the evolution of the business support systems (BSS) solutions for supporting 5G and the IoT. Besides 5G and the IoT, he has also interested in cloud computing and machine learning.

JORGE ROBERTO CARNEIRO is the Telecommunications Manager by SENAC-RJ, a member of CREA-RJ, CRA-RJ, and APJERJ, the Specialist in project management by UCM-RJ, a Specialist in networks and telecommunications systems engineering by the National Institute of Telecommunications (Inatel). He is currently developing the M.Sc. dissertation in telecommunications with the Inatel, focusing on governance and operation models for smart places for current and future technologies (NovaGenesis). He contributes to the Research and Development area with the ICT Laboratory, Inatel, as a Researcher in cyberinfrastructure, future internet architectures, and smart places. He is responsible for the ICT Laboratory Project Office. He has 25 years of expertise in ICT industry.

VITOR ALEXANDRE CAMPOS FIGUEIREDO received the bachelor's degree in computer engineering from the Federal University of São Carlos (SP), in 2000, and the master's degree in business intelligence from PUC-RJ, in 2002. He is currently pursuing the master's degree in telecommunications with the IoT Research Group's Laboratory with INATEL, Santa Rita do Sapucaí, Brazil. He is a specialist in web software development, SOA, and database with more than 15 years of experience in the telecommunications market.

JOEL J. P. C. RODRIGUES (S'01–M'06–SM'06) is currently a Professor with the Federal University of Piauí, Brazil, and a Senior Researcher with the Instituto de Telecomunicações, Portugal. He is the Leader of the Internet of Things Research Group (CNPq), the Director for Conference Development-IEEE ComSoc Board of Governors and the IEEE Distinguished Lecturer, the Past-Chair of the IEEE ComSoc Technical Committee on eHealth and the IEEE ComSoc Technical Committee on Communications Software, and a Steering Committee Member and the Publications Co-Chair of the IEEE Life Sciences Technical Community. He has authored or coauthored over 780 articles in refereed international journals and conferences and three books. He holds two patents. He is a licensed Professional Engineer (as Senior Member), a member of the Internet Society, and a Senior Member of the ACM. He has one ITU-T Recommendation. He was awarded several Outstanding Leadership and Outstanding Service Awards from the IEEE Communications Society and several best papers awards. He has been the general chair and the TPC chair of many international conferences. He is the Editor-in-Chief of the *International Journal on E-Health and Medical Communications* and editorial board member of several high-reputed journals.

• • •