Beyond Transmission Performance: 5G Opportunities for Developing e-Energy Applications*

Riccardo Rapuzzi
CNIT, S3ITI Lab
Genoa, Italy
riccardo.rapuzzi@cnit.it

Matteo Repetto
CNIT, S3ITI Lab
Genoa, Italy
matteo.repetto@cnit.it

Danilo Tigano
University of Genoa, DITEN
Genoa, Italy
danilo.tigano@unige.it

ABSTRACT

The advent of 5G networks is expected to bring radical changes in several key vertical sectors, including energy. Beyond the expected improvement of key performance indexes (like bandwidth, delay, jitter), the 5G concept also envisages a transformation of legacy communication infrastructures into intelligent orchestration platforms, which boosts new models and paradigms for service development and deployment.

In this paper, we shed light on some of the most overlooked key performance indexes and elaborate on the opportunity for quicker service development and deployment, by giving insight on the latest relevant emerging technologies in the cloud and network function virtualization fields.

KEYWORDS

Software development and deployment, 5G, Internet of Things, eEnergy applications

1 INTRODUCTION

The transition of the traditional power system towards the smart grid has already started, and entails several phases. Today, focus has been given primary to wide-area connectivity and high performance, targeting bandwidth, latency, availability, density, and reliability levels in mobile networks similar to wired infrastructures [10, 30]. Tomorrow, a large base of intelligent devices with processing and communication capability ("smart Things") will create the so called *Internet of Things – IoT*) and will make possible re-engineering a broad range of applications for effective control and management of the grid, as well as new business services.

The big challenge for this evolutionary process will be seamless and effective integration of Things into applications and services, tackling the multiplicity of devices and their heterogeneity, while making the software flexible, portable, and adaptable to different environments (for instance, the same control application should be usable in different microgrids without requiring modifications, and with minimal configuration and installation efforts). IoT middleware and frameworks have already been largely investigated, including context-brokers, service-oriented architectures, and other paradigms for decoupling information and context from physical devices [21, 23]. This facilitates software development but not service deployment: in fact, there is still the need for manual configuration every time the service is used in a different contexts or just for life-cycle management (e.g., in case of failures). The need for more elasticity and adaptability in deploying such applications has already been recognized recently by on-going research projects [7] and this approach has already been followed by the recently-launched SmartSDK project [26], which aims at becoming the FIWARE's "cookbook" for developing smart applications.

The importance of *descriptive* models in software development, which enable semi-autonomous applications for deployment and life-cycle management, has already been largely recognized. *Virtualization* and *orchestration* are the buzzwords that are recently re-shaping the software industry, starting from cloud applications and already involving networks (i.e., network function virtualization – NFV). The next step will be the extension to the IoT, and the 5G initiative has already anticipated this trend by some visionary concepts at first [27] and resulted in more concrete architectural layers recently [1]. Indeed, *service deployment time* is already included among the 5G Key Performance Indicators [1], but most telecommunications vendors and operators have largely overlooked it till now.

Future telecommunication networks can be thought as capillary fabrics of heterogeneous resources with computing, storage, and networking capability, building a *computing continuum* which extends from the cloud to the things [27]. Operation and management of such infrastructures will mostly be software-driven, opening the opportunity to host vertical services in addition to pure network function virtualization. Though the 5G vision may appear rather ambitious to some, several enabling technologies are already available and the convergence of things, networks, and applications over a common ICT fabric is not really a chimera.

In this paper, we elaborate on the great opportunity brought by the 5G architecture to radically change the way software applications are developed and deployed in the energy sector, by providing a quick but rather exhaustive understanding of recent virtualization and orchestration paradigms, with specific examples of their usage for developing control and management applications for the electrical grid. The rest of the paper is organized as follows. Section 2 gives insight on the 5G architecture, with focus on those layers specifically conceived to support vertical industries. Section 3 introduces recent development models and their usage to automate deployment and life-cycle management. Building on these technological prerequisites, Section 4 explains how smart things could be virtualized in the development process and automatically linked to physical devices later at service deployment time. Finally, we give conclusions in Section 5.

*Submitted version. The final publication is:
Riccardo Rapuzzi, Matteo Repetto, and Danilo Tigano. 2018. Beyond Transmission Performance: 5G Opportunities for Developing e-Energy Applications. In *e-Energy '18: International Conference on Future Energy Systems, June 12–15, 2018, Karlsruhe, Germany.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3208903.3212039

2 THE 5G VISION AND CONCEPTUAL ARCHITECTURE

At first glance, fifth-generation mobile networks (5G) may just look a straight evolution bringing key performance indexes one step further than today (e.g., radio access speed and latency, coverage and connected devices, data volumes, etc.), but there are indeed far more under the hood. While radio performances undeniably represent a fundamental aspect, 5G will largely build on recent advances in virtualization, targeting more flexibility and cost-effectiveness in network operation and management. As a matter of fact, next-generation telecommunication infrastructures will consist of a rich ICT fabric, with capillary computing, storage, and networking resources creating pervasive virtualization environments, where anything or everything may be offered as-a-service (XaaS) [27]. Network Function Virtualization (NFV) exploits this new paradigm to transform hardware appliances (firewalls, routers, base stations, packet cores, etc.) into software functions and to compose them in virtualized services, which are then deployed and orchestrated over the underlying virtualization infrastructure.

Though NFV by itself represents a landmark in the evolutionary path of telecommunication networks, it is just the tip of the iceberg. Indeed, these networks are the critical infrastructures to connect smart things: once computing and storage are capillary available, they will become a pervasive computing continuum with associated low latency, reliability, and Quality of Service (QoS) features. In current cloud-based paradigms, the distance to the things and the impossibility to manage dynamic service level agreements for the whole communication path turn into latency and bandwidth constraints, which prevent the realization of real-time and interactive services. Differently from the cloud, where resources are located in one or a few big data centers, 5G infrastructures will consist of large distributed, pervasive, heterogeneous, and multi-domain environments, with both traditional data centers and a large number of small/tiny edge installations. The combination of edge computing and network function virtualization will enable dynamic allocation of computing and storage resources to run software functions wherever needed, while providing end-to-end control and data plane connectivity between software peer entities and physical devices/terminals (in technical slang indicated as network *slicing*), in order to achieve the target end-to-end service performance.

5G yearns therefore to become the large-scale de facto standard interface between things and applications. As a matter of fact, with the expected grow in the installed base of things [2, 25], the massive amount of sensors, actuators, and data generated will require new paradigms beyond legacy middleware and cloud processing, which boost distributed processing and storage, low-latency communication, and more efficient usage of network bandwidth [28, 33]. 5G will evolve the network infrastructure from mere streams of bits to full virtualization and intelligent orchestration platforms, overlooking software deployment, connectivity, and data exchange [1].

The above vision is reflected in the layered 5G integrated architecture for mobile broadband and vertical services, depicted in Fig. 1. It may be viewed as the stack of network-centric services (at the lowest layers) and vertical-centric services (at the topmost layers).

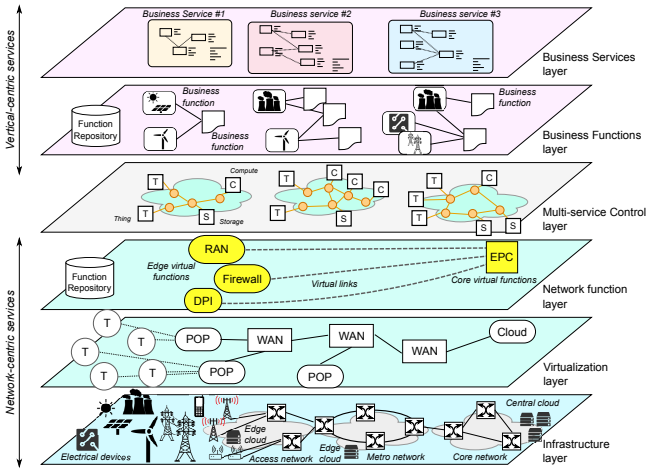

Figure 1: Integrated 5G architecture for mobile broadband and vertical services.

The *infrastructure layer* collects all physical resources, including switching/routing devices and communication links, massive computing and storage resources, smart things (sensors, actuators, gateways, smartphones, drones, robots, cars, etc.). It represents the ICT fabric, where things, computing, storage, and networking resources are interconnected by a broad range of access and transport technologies (optical, copper, radio and satellite links). The infrastructure will typically belong to different providers, which set up complex business relationships.

Resources at the infrastructure layer are then abstracted and exposed to upper layers in the *virtualization layer*. Here a mix of consolidated and upcoming technologies cluster resources together, offering virtual resources in terms of virtual machines, block and object storage (POP, Cloud), software-defined connectivity (WAN), logical IoT functions (T). This layer maps virtual instances to physical resources, hiding low-level management details to upper layers. Existing tools that may be used include cloud management software (e.g., OpenStack, vSphere, CloudStack) and SDN controllers (e.g., OpenDayLight, Ryu, NOX, commercial products from NEC and Brocade). Additionally, some effort has been undertaken to abstract, model, and "program" things (e.g., ThingML [17]).

The *network function layer* implements network services as combination of virtual functions. The core elements of this layer entail the repository of network virtual functions (firewall, encryption, Network Address Translation, load balancer, Radio Access Network, Deep Packet Inspection, Enhanced Packet Core, etc.), and the management mechanisms that provision virtual resources from the underlying layer, deploy the service, and orchestrate it during its life-cycle. It is worth noting that virtual functions may be deployed on general-purpose computing resources (i.e., virtual machines) or on dedicated hardware, whenever performances matter. There are several complementary frameworks that cope most of the technical aspects mentioned above: ETSI Network Function Virtualization [12], ETSI Mobile Edge Computing [13], IEEE Service Function Chaining [15], not to mention the plethora of research and industrial initiatives.

The *multi-service control layer* enables the creation, operation, and control of multiple separated communication networks, by 'slicing' the physical infrastructure. This will provide a dedicated network and all management functions to the service tenant, functionally indistinguishable by a real network, which interconnects things, computing and storage resources. In other words, this layer provides virtual networked execution environments for business services. This layer maps business requirements into network topology and configuration, by reserving physical resources and composing network services through the underlying network function layer. Key aspects of this layer are strict isolation among different tenant and enforcement of QoS according to contractual service level agreement.

The *business function layer* collects software functions for the vertical industries. It is somehow conceptually similar to the network function layer, but tailored to business services. Business functions include virtual instances of things, representing their capability and interfaces (e.g., power meter, phasor measurement, circuit breaker), and software functions, which implement elementary or complex operation (e.g., collecting meter readings and phasor measurements, opening or closing circuit breakers, load shedding, load and production forecast, demand-response schemes, failure and short-circuit detection). Such functions are then mapped to computing resources and things by the multi-service control layer, according to specific placement and QoS requirements in terms of latency, throughput, jitter, availability, security, etc. The paradigm entailed by this layer is the primary driving force for re-shaping electrical applications, so we'll elaborate more on this topic in the following Sections.

Finally, the *business service layer* describes value chains by composing generic and specific service functions for each industry. The description combines different activities, as logical sequences or conditional alternatives, while setting execution constraints for the underlying layers (e.g., accuracy, due dates, service levels, security and safety requirements). The whole set of requirements will impact both the selection of specific function implementation (business function layer) as well as the topology and configuration of the network slice (multi-service control layer).

3 SOFTWARE ORCHESTRATION AND DEPLOYMENT

Growing complexity and ever shorter product lifetimes are pushing a transition of many industries from hardware-intensive to software-driven systems, accelerating innovation and cutting down development and operation costs [20]. Cloud computing already provides elastic and cost-effective execution environments, but effective implementation and maintenance of large applications also seek more automation in software deployment and lifecycle management.

A transition from "prescriptive" (i.e., procedural languages) to descriptive models is already ongoing, both for cloud applications [14, 16, 18, 32] and network function virtualization [6, 19, 24]. Models describe the application as a logical topology of elementary microservices (which are virtual network functions in the NFV world) [5, 11].

Each microservice is an independent software unit, which implements a specific function; for instance, Apache (implementation)

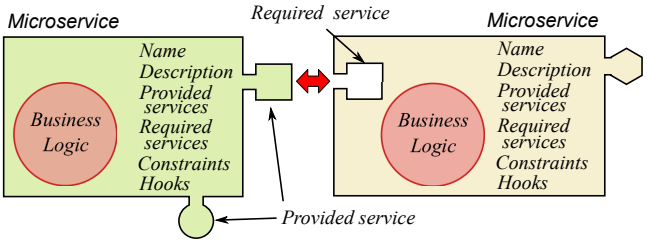

Figure 2: Two microservices can be linked together only if they are complementary (i.e., one microservice provides a service that the other requires).

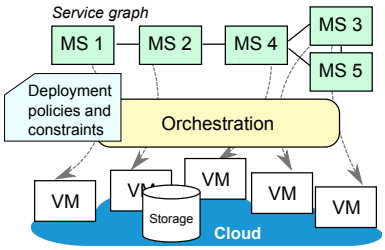

Figure 3: Service graph model and deployment over virtualized infrastructure.

web server (function), Bind9 (implementation) domain name server (function), and so on. A microservice is composed by a kernel part, which is the implementation of the business logic (i.e., the software) and a shell that contains metadata intended for automatic deployment and orchestration tools (see Fig. 2). Metadata include the name of the component (e.g., Apache web server, Bind9, WordPress), its description (including licensing and usage terms), provided functionality (e.g., HTTP server, DNS), required services (e.g., mysql database, file server), deployment constraints (e.g., number of cores, CPU speed, RAM, disk space, network bandwidth), measured performance metrics, and management hooks (for instance, to start, stop, reload, or reset the service, to collect measurements, data, events, log).

A service graph is the logical topology that interconnects microservices (Fig. 3). Each node is a microservice, and each link is a logical connection. A logical connection represents some kind of client/server relationship, communication link, or other dependency; it implies specific configuration actions at deployment time (e.g., IP address of the server, username and password, encryption secrets). Two microservices can be connected together only if one of them provides a service that is requested by the other (for example, we can connect a mysql client with a mysql server). Graphs also include deployment constraints and management policies. Policies are typically if-then statements that describe management actions upon verification of the specific condition; for instance, a component may be replicated when its workload rises above a given threshold[1], or a slave component may be activated when the master fails.

[1]This operation is called 'horizontal' scaling.

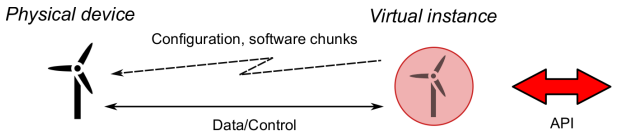

Figure 4: Virtual instances are logical abstractions of physical devices that hide all communication and configuration details.

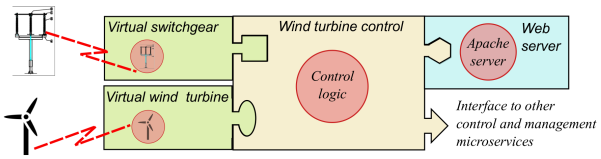

Figure 5: A simple example of control applications for electrical devices.

Some standards already exist that define the syntax and semantics for microservices and services graphs [12, 31]; there are also software orchestration tools that adhere to such standards or implements alternative or enhanced solutions [3, 8, 22, 29]. Thanks to the model-based approach and the usage of graphical interfaces, implementing an application is as simple as dragging and dropping microservices to the dashboard, drawing links, and setting properties [4, 9]. Then, the orchestration process takes a descriptive service graph and, according to its policies and metadata, selects proper implementation for each microservice from available software repositories, provisions virtual resources from virtualization environments, deploys and configures the software, monitors software execution, and performs management actions.

4 INTEGRATING ELECTRICAL DEVICES INTO 5G

The software development and orchestration paradigms briefly outlined in Sec. 3 represent key technological enablers to implement the topmost layers of the 5G architecture, but the creation of innovative applications in the energy vertical industry still requires a fundamental tile: things.

We argue that things may be easily integrated in emerging software paradigms by creating virtual instances. A virtual instance is an abstraction of a real object that abstract all configuration and communication details with the real world (TCP/UDP sockets, protocols, authentication, etc.), while exposing simple Application Programming Interfaces (APIs) for the logical functions the device offers (e.g., read measurements, write configuration parameters, move or rotate some mechanical parts), as shown in Fig. 4. Different instances may be used to provide different services: for example, a simple implementation for a wind turbine may give read-only access to operational parameters (like rotation speed, energy generation, frequency and voltage measurement), while another implementation may give raw access to IEC 61850 or similar protocols. Smart things will also have computing capabilities for carrying on low-latency tasks (*fog computing*), hence virtual instances will be also responsible to upload software chunks where necessary. Clearly, binding virtual instances with specific physical devices is not trivial, but this task will be managed by the 5G architecture (virtualization and business functions layers) through registration and discovery functions similar to those already used for cloud installations.

By wrapping virtual instances with specific metadata and service hooks, we get an orchestrable microservice that can be linked to other software components. To better explain the concept, we can think microservices as semantically equivalent to Java interfaces or C++ pure abstract classes, while virtual instances are specific implementations selected at deployment time according to required physical devices. The purpose of virtual instances is very similar to IoT middleware (e.g., the FIWARE eco-system). However, microservices are more easily and seamlessly plugged into applications, while the orchestration process automatically manages all configuration, deployment, and lifecycle issues.

Fig. 5 depicts a very simple example. We define specific interfaces for a renewable generator (ellipse shape) and for an electrical switchgear (square shape). The APIs for wind turbine provides access to operational parameters (like rotation speed, voltage, current, frequency) and reports failures, errors, and other logs. The API for the virtual switchgear reports current position and allows to change the position (open/closed). The developer of the control logic for the wind turbine uses simple function calls, just like the two devices were an internal routine, without the need for more complex network APIs (e.g., RESTful XML or JSON protocols) or direct use of TCP/UDP sockets. The interface between the turbine control and the web server is again a simple function call, which returns an HTML or XML formatted page that will be transferred to the client by the web microservice. Even if we cannot go into more technical details for the sake of brevity, we again remark that most of time-consuming configuration issues can be triggered and carried out by orchestration engines, while the developer focuses on the application topology and the selection of the field components that must be used.

Developing the above service would be as simple as dragging and dropping the microservices on the dashboad, drawing links among them, and set deployment requirements and policies. With the proper constraints on latency, the 5G orchestrator will take care of finding the right set of physical devices, selecting proper microservices implementations, and deploying low-latency tasks at the edge cloud closest to physical devices, while a dedicated network slice will be instantiated for communication. If we need the same system to control another turbine at another location, we can just copy the service graph and edit its properties, by selecting different physical devices. The whole process would require just a few minutes, to draw the graph and set the properties, while a conventional approach would require days or weeks to properly configure the hardware appliances and the electrical wiring.

5 CONCLUSIONS

In this paper, we have briefly reviewed the big potential behind the 5G concept, beyond the main key performance indicators. With capillary and pervasive computing, storage, and networking resources, 5G long for becoming a de-facto IoT framework with guaranteed performances for real-time services and applications.

Though we have not discussed in details all technical aspects and practical implications behind the proposed concept (for example about security and shareability of the components), we think that our work sheds light on a very important research topic. We plan to go on with the definition of the virtual instances for electrical appliances by developing working microservices for electrical devices in a microgrid, and to demonstrate and evaluate them in real-field experimental testbeds, which are now being developed in Europe in the context of the 5G PPP.

ACKNOWLEDGMENTS

The work was supported in part by the European Commission under Grant No.: 761898 (http://www.matilda-5g.eu/).

REFERENCES

[1] 5G PPP 2016. 5G empowering vertical industries. Whitepaper from the 5G-PPP, ERTICO, EFFRA, EUTC, NEM, CONTINUA and Networld2020 ETP. Retrieved December 14th, 2017 from https://5g-ppp.eu/wp-content/uploads/2016/02/BROCHURE_5PPP_BAT2_PL.pdf

[2] ABIresearch 2013. More Than 30 Billion Devices Will Wirelessly Connect to the Internet of Everything in 2020. Press release. Retrieved January 10th, 2018 from https://www.abiresearch.com/press/more-than-30-billion-devices-will-wirelessly-conne/

[3] ARCADIA 2015. A Novel Reconfigurable By Design Highly Distributed Applications Development Paradigm Over Programmable Infrastructure. EU H2020 programme, project no. 645372. Retrieved January 23rd, 2018 from http://www.arcadia-framework.eu

[4] ARCADIA 2016. Exploring the ARCADIA orchestrator for building and managing highly distributed applications. Video. https://www.youtube.com/watch?v=_Z8fLteTsK0

[5] Armin Balalaie, Abbas Heydarnoori, and Pooyan Jamshidi. 2016. Microservices Architecture Enables DevOps: Migration to a Cloud-Native Architecture. *IEEE Software* 33, 3 (March 2016), 42–52.

[6] Paolo Bellavista, Luca Foschini, Riccardo Venanzi, and Giuseppe Carella. 2017. Extensible Orchestration of Elastic IP Multimedia Subsystem as a Service Using Open Baton. In *5th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)*. San Francisco, CA – USA, 88–95.

[7] bIoTope 2016. *Edge Data Storage and Intelligent Filtering*. Technical Report. Retrieved March 2nd, 2017 from http://api.ning.com/files/klx1aEk42WqPkUYfdQu9joEUwwlC-aEseEWyvB0ytSj*WyjGgRTIwMYbPOaLTkPsiFmGEyaOz-jNHNbh-wjmQPSg5-FOYXS1/bIoTope_D4.1_Edge_Data_Storage_and_Intelligent_Filtering_Framework_v1.0.pdf bIoTope project.

[8] Canonical Ltd. 2018. Juju orchestrator. Web site. Retrieved January 17th, 2018 from https://jujucharms.com

[9] Jorge O. Castro. 2017. Juju GUI 2.0.3 released. Video. https://www.youtube.com/watch?time_continue=38&v=--flEVHP9gg

[10] Cisco 2016. Cisco 5G Vision Series: Vertical Value Creation. Whitepaper. Retrieved December 22nd, 2017 from https://www.cisco.com/c/dam/en/us/solutions/collateral/service-provider/ultra-services-platform/5g-vision-series-vertical-value-creation.pdf

[11] Nicola Dragoni, Saverio Giallorenzo, Alberto Lluch Lafuente, Manuel Mazzara, Fabrizio Montesi, Ruslan Mustafin, and Larisa Safina. 2016. Microservices: yesterday, today, and tomorrow. Cornell University Library. https://arxiv.org/abs/1606.04036v1 arXiv:1606.04036v1.

[12] ETSI MANO 2014. Network Functions Virtualisation (NFV); Management and Orchestration. ETSI GS NFV-MAN 001. Retrieved February 13th, 2017 from http://www.etsi.org/deliver/etsi_gs/NFV-MAN/001_099/001/01.01.01_60/gs_nfv-man001v010101p.pdf V1.1.1.

[13] ETSI MEC 2016. Mobile Edge Computing (MEC); Framework and Reference Architecture. ETSI GS MEC 003. Retrieved April 20th, 2017 from http://www.etsi.org/deliver/etsi_gs/MEC/001_099/003/01.01.01_60/gs_MEC003v010101p.pdf V1.1.1.

[14] Panagiotis Gouvas, Constantinos Vassilakis, Eleni Fotopoulou, and Anastasios Zafeiropoulos. 2016. A Novel Reconfigurable-by-Design Highly Distributed Applications Development Paradigm over Programmable Infrastructure. In *Proceedings of the 28th International Teletraffic Congress (ITC 28)*. Wuerzburg, Germany, 7–12.

[15] J. Halpern and C. Pignataro. 2015. Service Function Chaining (SFC) Architecture. RFC 7665. Retrieved May 12th, 2017 from https://tools.ietf.org/rfc/rfc7665.txt

[16] Rui Han, Moustafa M. Ghanem, and Yike Guo. 2013. Elastic-TOSCA: Supporting Elasticity of Cloud Application in TOSCA. In *The Fourth International Conference on Cloud Computing, Grids and Virtualization*. Valencia, Spain, 93–100.

[17] Nicolas Harrand, Franck Fleurey, Brice Morin, and Knut Eilif Husa. 2016. ThingML: A Language and Code Generation Framework for Heterogeneous Targets. In *Proceedings of the ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems (MODELS '16)*. Saint-malo, France, 125–135.

[18] Bill Karakostas. 2014. Towards Autonomic Cloud Configuration and Deployment Environments. In *International Conference on Cloud and Autonomic Computing (ICCAC)*. London, UK, 93–96.

[19] Ahmed M. Medhat, Tran Quang Thanh, Stefan Covaci, Giuseppe Carella, and Thomas Magedanz. 2016. Orchestrating Service Function Chaining in Cloud Environments. In *IEEE Sixth International Conference on Communications and Electronics*. Ha Long, Vietnam, 532–538. Poster session.

[20] NESSI 2017. Software Engineering – Key Enabler for Innovation. Whitepaper. Retrieved February 28th, 2017 from http://www.nessi-europe.eu/Files/Private/NESSI_SE_WhitePaper-FINAL.pdf

[21] A. H. Ngu, M. Gutierrez, V. Metsis, S. Nepal, and Q. Z. Sheng. 2017. IoT middleware: a survey on issues and enabling technologies. *IEEE Internet of Things Journal* 4, 1 (February 2017), 1–20.

[22] OpenBaton 2017. An open-source reference implementation of ETSI NFV MANO framework. Web site. Retrieved December 15th, 2017 from https://openbaton.github.io

[23] M. A. Razzaque, M. Milojevic-Jevric, A. Palade, and S. Clarke. 2016. Middleware for Internet of Things: a survey. *IEEE Internet of Things Journal* 3, 1 (February 2016), 70–95.

[24] Jordi Ferrer Riera, Josep Batallé, José Bonnet, Miguel Días, Michael McGrath, Giuseppe Petralia, Francesco Liberati, Alessandro Giuseppi, Antonio Pietrabissa, Alberto Ceselli, Alessandro Petrini, Marco Trubian, Panagiotis Papadimitrou, David Dietrich, Aurora Ramos, Javier Melián, George Xilouris, Akis Kourtis, Tasos Kourtis, and Evangelos K. Markakis. 2016. TeNOR: Steps towards an orchestration platform for multi-PoP NFV deployment. In *IEEE NetSoft Conference and Workshops (NetSoft)*. Seoul, South Korea, 243–250.

[25] J. Rivera and R. van der Meulen. 2013. Gartner Says the Internet of Things Installed Base Will Grow to 26 Billion Units By 2020. Press release. Retrieved April 1st, 2017 from http://www.gartner.com/newsroom/id/2636073

[26] SmartSDK 2016. A FIWARE-based SDK for developing Smart Applications. EU's Horizon2020 project, no. 723174. Retrieved January 23rd, 2018 from https://www.smartsdk.eu

[27] D. Soldani and A. Manzalini. 2015. On the 5G Operating System for a True Digital Society. *IEEE Vehicular Technology Magazine* 10, 1 (March 2015), 32–42.

[28] J. A. Stankovic. 2014. Research directions for the Internet of Things. *IEEE Internet of Things Journal* 1, 1 (February 2014), 3–9.

[29] T-NOVA 2017. TeNOR. Software repository. Retrieved January 23rd, 2018 from https://github.com/T-NOVA/TeNOR

[30] Linus Thrybom and Ádám Kapovits. 2015. 5G and Energy. Whitepaper. Retrieved January 23rd, 2018 from https://5g-ppp.eu/wp-content/uploads/2014/02/5G-PPP-White_Paper-on-Energy-Vertical-Sector.pdf Version 1.0.

[31] TOSCA 2013. Topology and Orchestration Specification for Cloud Applications. OASIS Standard. Retrieved March 20th, 2017 from http://docs.oasis-open.org/tosca/TOSCA/v1.0/os/TOSCA-v1.0-os.pdf Version 1.0.

[32] J. Wettinger, U. Breitenbücher, and F. Leymann. 2014. Standards-Based DevOps Automation and Integration Using TOSCA. In *IEEE/ACM 7th International Conference on Utility and Cloud Computing (UCC)*. London, UK, 59–68.

[33] Ben Zhang, Nitesh Mor, John Kolb, Douglas S. Chan, Nikhil Goyal, Ken Lutz, Eric Allman, John Wawrzynek, Edward Lee, and John Kubiatowicz. 2015. The Cloud is Not Enough: Saving IoT from the Cloud. In *7th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 15)*. Santa Clara, CA – USA.