An Edge Operating System Enabling Anything-as-a-Service

The authors argue that SDN and NFV, together with cloud and edge-fog computing, can be seen as different facets of a systemic transformation of telecommunications and ICT, called softwarization. The first impact will be at the edge of current telecommunications infrastructures, which are becoming powerful network and service platforms. The edge operating system (EOS) software architecture is proposed as the means to get there.

Antonio Manzalini and Noel Crespi

Abstract

This article argues that SDN and NFV, together with cloud and edge-fog computing, can be seen as different facets of a systemic transformation of telecommunications and ICT, called softwarization. The first impact will be at the edge of current telecommunications infrastructures, which are becoming powerful network and service platforms. The edge operating system (EOS) software architecture is proposed as the means to get there. In fact, the main feature of EOS is to bring several service domains, such as cloud robotics, Internet of Things, and Tactile Internet, into convergence at the edge. The development of EOS leverages available open source software. A use case is described to validate the EOS with a proof-of-concept.

COMMUNICATIONS STANDARDS

Context and Drivers

We are witnessing a period of rapidly growing interest on the part of industry and academia in software-defined networks (SDN) [1] and network function virtualization (NFV) [2]. The growing interest in these paradigms (re-proposing principles have been well known) is most probably motivated by the novelty of the overall context, specifically their techno-economic sustainability and high-level performance. These advances are mainly due to the technological milestones achieved in the last two decades: the impressive diffusion of fixed and mobile ultra-broadband, the increasing performance of chipsets and hardware architectures, the ever-growing availability of open source software, and the cost reductions (determined also by a shift in how IT services are provided).

This article argues that, thanks to these techno-economic trends, SDN and NFV principles will soon impact not only current telecommunications networks, but also service and application platforms. In fact, SDN and NFV, together with cloud, edge and fog computing, can be seen as facets of a broad innovation wave (called softwarization) that is accelerating the ongoing migration of "intelligence" toward the users.

In view of that, it is argued that the first impact of softwarization will be at the edge, which is defined as the peripheral part of current infrastructures, ranging from the distribution and access segments up to the direct proximity to users (e.g., home, office).

While cloud computing is a well known paradigm, already exploited from an industrial point of view, the concepts of edge and fog computing require, at least for this article, a short definition. Concerning the former, we basically refer to ETSI [3] which defines mobile edge computing as the method of providing IT and cloud-computing capabilities within the radio access network (RAN) in close proximity to mobile subscribers. Fog computing pushes the edge computing paradigm up to the end users terminals (e.g., smart phones) and other devices, which will be able to store pieces of data and to execute service components locally.

Softwarization will be a radical change of paradigm. Current telecommunications infrastructures have been exploited with purpose-built equipment designed for specific functions. In the future, network functions and services will be virtualized software processes executed on distributed horizontal platforms mainly made of standard hardware resources.

Standard hardware and open source software will play a strategic role in this profound transformation, by fuelling open innovation while reducing the investments required to deploy said infrastructures. For example, OpenStack [4] is an open source platform designed to provide cloud services. Several pre-standardization bodies and fora regard OpenStack as an ideal candidate for developing orchestration features in NFV infrastructures. Other examples of open source software are the SDN controllers that have been released to date, and ONOS [5].

SDN and NFV open source software adoptions will fuel innovation and reduce software costs. This does not not necessarily mean CAPEX and OPEX reductions for operators and service providers. In fact, most open source software products may eventually require contracted third party support to become exploitable in production environments for commercial, industrial, financial, and public service applications.

On the other hand, the entire value chain will change radically: the "thresholds" for new players to enter the telecommunications and ICT (Information and Communication Technology) markets [6] will be lowered.

Cost savings alone will not be enough to assure the future sustainability of the telecommunications industry: it is key also to enable innovative service paradigms. Two often-mentioned examples are "immersive communications" and "anything as a service," service paradigms that are posing challenging requirements for future telecommunications infrastructures.

"Immersive communications" looks beyond the "commoditization" of current communication paradigms (e.g., voice, messaging, etc.) by addressing new advanced forms of social communications and networking (e.g., artificially intelligent avatars, cognitive robot-human interactions, etc.).

"Anything as a service" is about providing (anytime and anywhere) wider and wider sets of ICT services by means of new terminals, even

Antonio Manzalini is with Telecom Italia, Strategy and Innovation — Future Centre.

Noel Crespi is with Institut Mines-Telecom.

going far beyond our imagination (e.g., intelligent machines, robots, drones, and smart things) [7]. Imagine, for example, services for improving industrial and agricultural efficiency, for enabling decentralized micro-manufacturing, for improving efficiency in private-public processes, and for creating and maintaining smart environments.

In summary, softwarization is a systemic transformation. It is not just about the introduction of another technology or network layer in current infrastructures. Rather, it goes beyond the networks to also impact the service platforms and the future role of terminals. In this respect, beyond the technological aspects, softwarization implies business sustainability and strategic regulatory issues.

The outline of this article is as follows. We outline the main enabling technologies, and we describe the software architecture of the edge operating system (EOS). We then describe a use-case and elaborate on the design and development of the EOS, leveraging open source software. Closing remarks are provided in the last section.

Enabling Technologies of Telco Softwarization

SDN and NFV are two of the most-discussed technologies capable of enabling the softwarization of telecommunications.

SDN relies on the separation of control and data-forwarding functions. In principle, this is applicable to any node of a telecommunication network (e.g., a switch, a router, or other transmission equipment). Another key characteristic of SDN is the possibility of executing the above-mentioned (control) software outside of the equipment boundaries, for example on dedicated IT servers or even in a data centre (e.g., cloud computing). Control programmability (via APIs) is a third relevant aspect of SDN.

NFV is about the virtualization of network functions and their dynamic allocation and execution on (almost) general purpose processors (e.g., x86), shared over multiple customers, datastreams, and applications. SDN and NFV are not directly dependent, but they are mutually beneficial. In fact, when coupled, they amplify their potential innovation impact on telecommunications infrastructures.

If software-hardware decoupling and the virtualization of functions and services can be seen as the "common denominator" of softwarization, the potential differentiation and evolution of cloud toward edge and fog computing represents other interesting and synergistic expressions of the same overall transformation.

TPC is a serious performance bottleneck for video and other large files (as it requires receiver acknowledgement) and throughput is inversely related to round trip time (RTT) or latency. It is impossible to provide HD-quality streams if the servers are not relatively close to the users. At the same time, with just best effort traffic it will not be possible to achieve the low latency requirements posed by services such as caching or interactive applications.

In fact, the "last mile" connection between a user and the ISP is a significant bottleneck. According to the FCC's Measuring Broadband America report [8], during peak hours "Fiber-to-the-home services provided 17 milliseconds (ms) round-trip latency on average, while DSL-based services averaged 44 ms."

It should be mentioned that PON and DSL delays are intrinsic in the access protocols. Achieving lower delays means either changing said protocols or locating all of the necessary data at the subscriber, including content caches and databases.

In the latter direction, fog computing pushes the edge computing paradigm even further, up to the end users' terminals and devices, which are storing data and locally executing pieces of service logic. This will further amplify the diffusion of applications and the migration of "intelligence" toward the users.

In summary, it is very likely that techno-economic drivers and emerging technologies will create the conditions for exploiting very powerful network and service platforms at the edges of current infrastructures. Such platforms will be able to carry out a substantial amount of storage and real time computation, thereby supporting a wide range of innovative communications and ICT services.

It is very likely that the techno-economic drivers and the emerging technologies will create the conditions for exploiting very powerful network and service platforms at the edges of current infrastructures. Such platforms will be able to carry out a substantial amount of storage and real time computation.

The Edge Operating System

The edge of current telecommunications infrastructures (i.e., the access areas up to the direct proximity to users) will become powerful network and service platforms. The EOS software architecture proposed by this article is the means to get there. EOS will provide the typical services of an operating system, e.g., abstractions, low-level element control, commonly-used functionalities, message-passing between processes, management of packets of processes, etc.

For the basic design of the EOS, we took our inspiration from the architecture of the robot operating system (ROS) [9], an open source, widely adopted meta-operating system for robotic systems. Among the merits of ROS that have been adopted by EOS is the variety of processes (called nodes), executed on a number of different hosts, connected at runtime with logical topologies.

Moreover, another main reason for that design choice is the observation that a robot, generally speaking, can be considered a dynamic aggregation of resources such as sensors, actuators, and processing-storage capabilities, implementing a cognitive loop. These are the same categories of resources that will populate the edges of current infrastructures, named infrastructure elements (IE).

Obviously the domain contexts ROS and EOS applications are quite different; in fact, the design of the EOS software architecture has been extended to meet the edge requirements. In particular, a physical IE (Fig. 1) has been defined to include any dynamic combination of sensors, actuators, processing-storage resources, and data forwarding capabilities. Sensors, actuators, robots, drones, routers, and terminals can all be seen as particular physical IEs. This generalization will help in structuring the functional model of the EOS.

From a functional perspective, IE will provide a set of services, leveraging the concept of the self-managed cell reported in [10]. For example, the set of services may include: discovery services

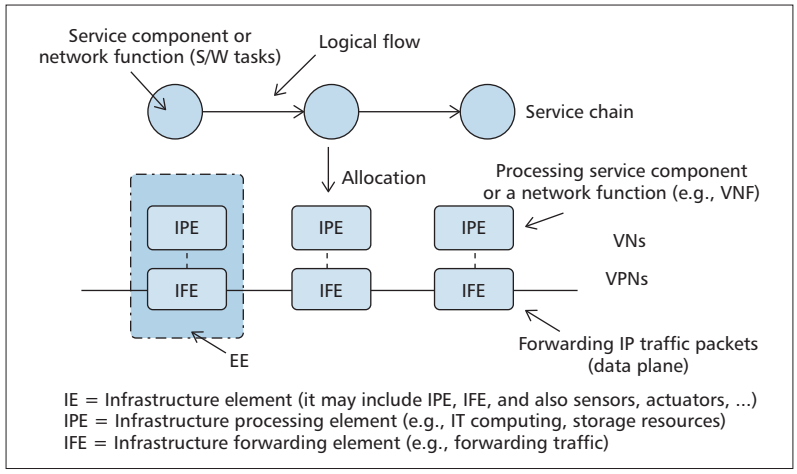

Figure 1. Edge elements.

to discover local resources and components as part of the physical IE; policy services to manage the policies specifying IE behavior; or even cognitive services to implement a certain level of cognition (Fig. 2), even more when coupled with sensing and actuating capabilities.

Generalizing, we can say that cognition (implemented through artificial intelligence methods, deep learning techniques, heuristics, algorithms, etc.) will allow IEs to learn and reason about how to behave in response to goals in a complex context, or at least be able to optimally execute their service and network functions.

Key Characteristics of the Robot Operating System

The ROS is a widely adopted meta-operating system for robots. The full source code of ROS is publicly available and currently runs only on Unix-based platforms. A robotic system built using the ROS consists of a number of processes (ROS nodes), potentially on a number of different hosts, connected at runtime in a peer-to-peer topology. ROS has a lookup mechanism (ROS master) to allow processes to find each other at runtime.

ROS master acts as a name-service, storing topics and service registration information for ROS nodes. Nodes communicate with the master to report their registration information. As these nodes communicate with the master, they can receive information about other registered nodes and make connections as appropriate (bypassing messages and structuring data).

Nodes connect to other nodes directly; the master only provides lookup information, much like a DNS server. Nodes that subscribe to a topic will request connections from nodes that publish that topic, and will establish that connection over an agreed upon connection protocol (e.g., standard TPC/IP sockets). This is represented in Fig. 3.

An ROS node sends a message by publishing it to a given topic, which is simply a string such as "map." A node that is interested in a certain kind of data will subscribe to the appropriate topic. There may be multiple concurrent publishers and subscribers for a single topic, and a single node may publish and/or subscribe to multiple topics.

Although the topic-based publish-subscribe model is a flexible communications paradigm, it is not appropriate for synchronous transactions, which can simplify the design of some nodes. Therefore, ROS developers have introduced the concept of services, defined by a string name and a pair of strictly typed messages, one for the request and one for the response. This is analogous to web services, which are defined by URIs and have request and response documents of well defined types.

The special characteristics of the ROS architecture allow for decoupled operation, wherein names are the primary means by which larger and more complex systems can be built. This decoupling is one of the main reasons why we have taken most of our inspiration from ROS when designing the EOS. One of the EOS requirements, in fact, is to allow the flexible and scalable operations of complex and dynamic systems, built by aggregations of IEs.

Assumptions of the Edge Operating System

EOS leverages the concept of services as representing a sort of "unifying" abstraction across physical edge resources, across multiple infrastructure domains, and across different service levels. A service provides a function (e.g., from ISO-OSI L2 to L7, so it could also be a network function, or a middle-box), it exports an API, it is available anywhere and anytime (location-time independent), is scalable, elastic, and resilient, and it can be composed with other existing S/W components (e.g., to create a service chain). Services are executed into one or more infrastructure virtual slices, which are made of a set of logical resources (e.g., virtual machines, containers) connected through virtual networks.

The allocation and orchestration of logical resources, in charge of executing a service chain, requires solving constraint-based double optimization problems. Not only do VMs have to be properly allocated (to avoid hot-spots), but also the traffic crossing the VMs has to be properly routed (to avoid congestion).

The term orchestration has long been used in the IT domain to refer to the automated tasks involved with arranging, managing, and coordinating higher-level services provisioned across different applications and enterprises [11]. In the SDN-NFV, orchestration is concerned with lower-level (i.e., network) services, with a comprehensive management of both IT and network logical resources.

EOS software adopts a publish-subscribe model (Fig. 3) [13] as a basic way to distribute software task execution requests. Each software task execution request is coded as a tuple and written on the tuple space, named blackboard, while a take operation is used by IEs to offer their process capability.

Functional Architecture of the Edge Operating System

This sub-section describes the main characteristics of the EOS, whose high-level architecture is reported in Fig. 4.

The main elements of the EOS are listed below.

• An EOS node is an S/W module that can be executed on top of any operating system (e.g., Linux-based OS, Android, Robot Operating System, etc.) of an IE. Similar to the ROS, any EOS node communicates with an EOS master to whom it registers (e.g., services that it can pro-

vide) and updates the status (e.g., resource utilization) of its associated IEs. This data is stored in the EOS master data base (EOS MDB). IE nodes are interconnected on the data plane via fixed and virtual radio links (these links could be either local, in a single edge domain, or across a WAN).

- An EOS master is dynamically allocated to a specified edge domain. It is responsible for the local creation (and deletion) of the slice(s) where service chains are executed in order to provide the requested services. It has to interact with a higher-level EOS orchestrator and with other EOS masters, in case the service chain has to be allocated across multiple edge domains. The EOS MDB stores the data related to the IEs belonging (assigned) to the specified domain.
- The master blackboard is a sort of virtual repository shared among the EOS master and the EOS nodes. The EOS master publishes (using the pub primitive) the task/component of the service chain that has to be allocated. In turn, EOS nodes subscribe (using the sub primitive) to the S/W task/component if the associated IE can provide the logical resources to execute it (i.e., can serve the specific task of the chain). Multiple subscriptions are possible, so in a next stage the EOS master will make an optimized selection of the IEs to whom the S/W task/component will eventually be allocated.
- The collector abstraction [12] has been introduced to make master blackboards recursive, thus overcoming scalability limitations. In this sense, a collector can be seen as an agent acting as an ensemble of IEs together with their shared blackboard. A collector thus can act toward other collectors as a super-IE, as it can take tasks on its blackboard from other overloaded collectors.
- The EOS master includes a capability called the selection method, which makes it possible to select the proper IEs to assign the execution of the service chain tasks. Selection is done according to specific criteria, for example through the minimization of specified KPI, as with end-to-end application latency. Interacting with the

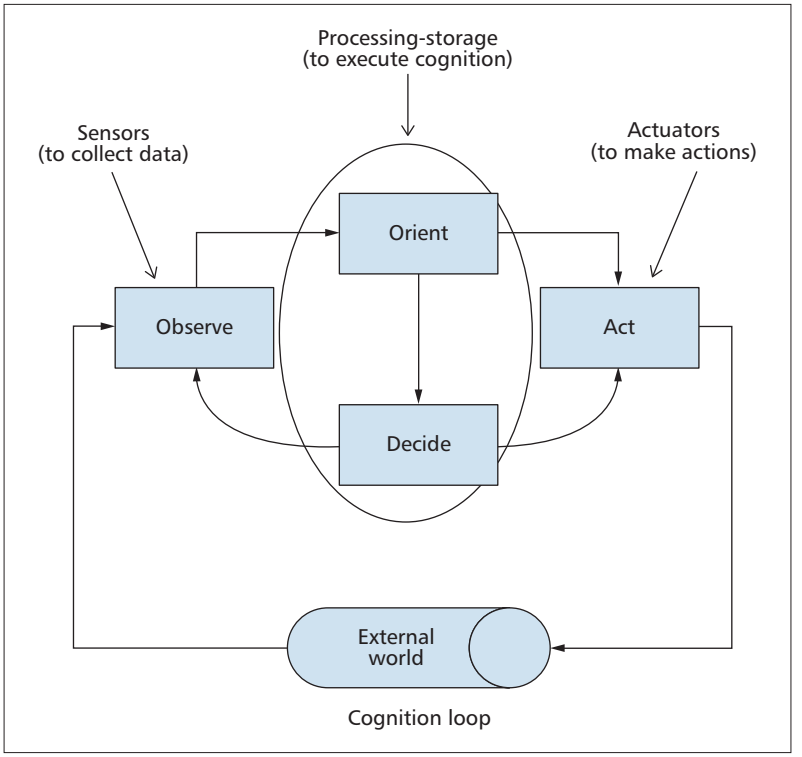

Figure 2. Cognitive loop implementable in an IE.

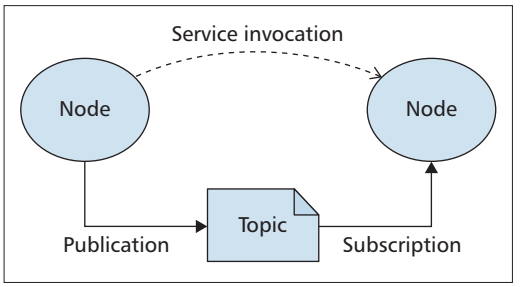

Figure 3. ROS: pub-sub model.

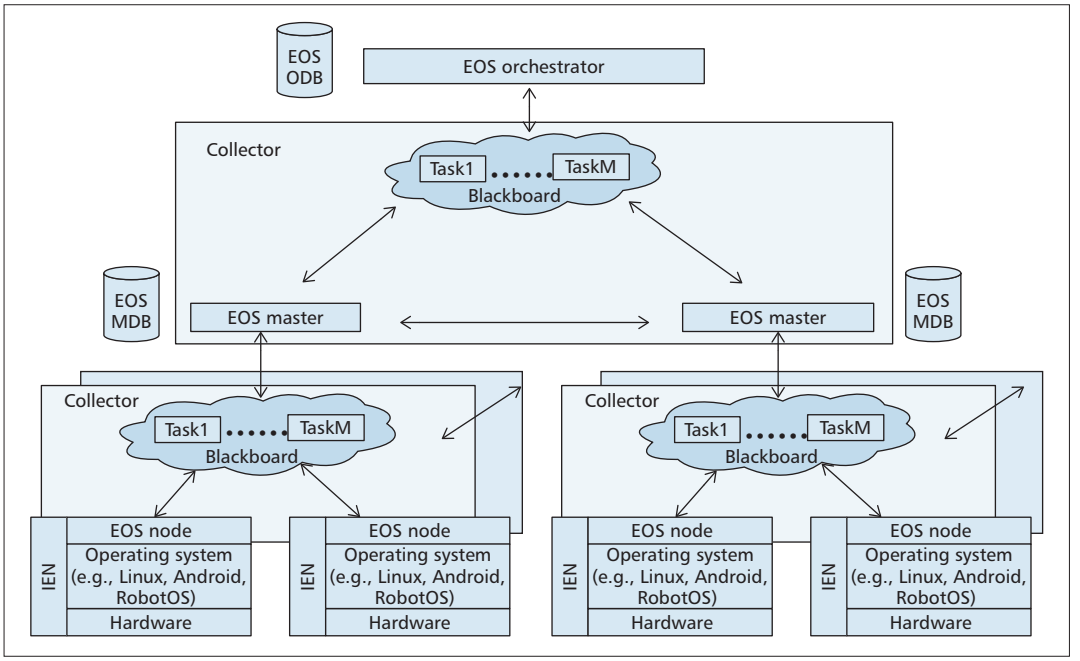

Figure 4. High-level functional architecture of the EOS.

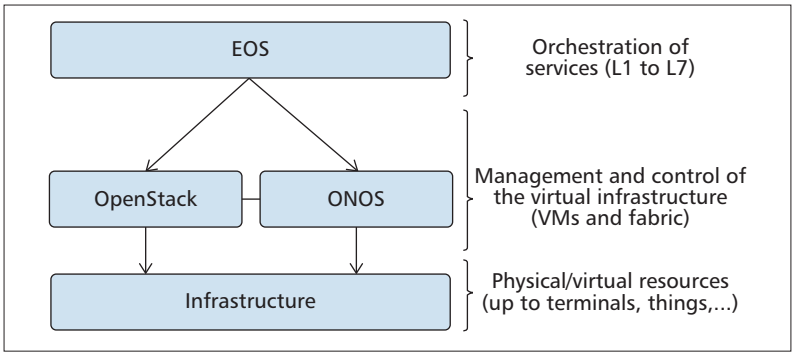

Figure 5. EOS prototype.

EOS node, the EOS master can configure the logical resources hosted by the IEs at run time.

- The EOS orchestrator is a higher-level S/W module that is set up to receive the service request that must span multiple edge domains. It decomposes the request in a service chain, selecting and interacting with the appropriate EOS masters for the end-to-end allocation of IEs across multiple domains. The EOS master can communicate with the EOS orchestrator to whom it registers and updates its status. This data is stored in the EOS orchestrator data base (EOS O-DB).
- The orchestrator blackboard is a higher-level virtual repository shared between the EOS orchestrator and the EOS masters. As above, for the master blackboard, the collector concept can be applied at this level also for the orchestrator blackboard.

The EOS software architecture can be seen as an expression of the integration of the SDN functional architecture defined in ONF (e.g, EOS nodes are like controllers) and the NFV reference architecture framework by ETSI NFV (e.g., an EOS master has VIM capabilities).

Next we briefly describe an example that shows the functioning of EOS. At the startup of an IE, the EOS node sends the EOS master a description of the associated IE services and the status (including the configuration) of the resources. The description can adopt a variety of formats, e.g., YANG modeling [13].

The users' service requests are sent to the EOS master through their terminals (which run EOS nodes). If a service request can be executed just locally, within the EOS master domain, then it is simply decomposed into a sequence of service components and required network functions (i.e., a service chain).

The EOS master then publishes (pub primitive) the software tasks of the service chain. EOS nodes subscribe (sub primitive) to said tasks if the related IEs can execute them. At the end, the EOS master must make an optimized selection of the IEs (selection method). On the other hand, if the EOS master realizes that the service request cannot be executed locally, it forwards it to the EOS orchestrator. In turn, the EOS orchestrator decomposes the service request and publishes it on its blackboard. The flow of actions then proceeds as above within each edge domain.

The EOS is a distributed software architecture where the states of the resources are store in distributed DBs. The well-known CAP theorem [14] will dictate some limitations. In fact, it states that any networked shared-data system can have at most two of the following three properties: 1) consistency (C) equivalent to having a single up-to-date copy of the data; 2) high availability (A) of that data (for updates); and 3) tolerance to network partitions (P).

The general idea is that two of the three properties have to be privileged (CP favors consistency, AP favors availability, and with CA there are no partitions), a trade-off that will be needed then for storing/configuring the states of the infrastructure while achieving specific performance levels. End-to-end latency (or delay) and partitioning are deeply related, and such relations become more important in the case of a widely distributed infrastructure. This situation contributes even more to the requirement of minimizing the application end-to-end latency. These areas require further investigation.

Use Case: Mobile Cognitive Machines

The use case described in this section aims at both the definition of main challenges and requirements for EOS and the feasibility demonstration of a prototype. The main concept of the use case is about the pervasive adoption of mobile cognitive machines (Fig. 2) provisioning any sort of ICT services.

Already today we are witnessing growing interest in using drones, robots, and autonomous machines in agriculture, industry, security, and several other domains. For example, the advent of robots remotely controlled via 5G connections would create a tremendous impact on Industry 4.0. Also, the contexts of the Tactile Internet and cyber physical systems envision several applications for cognitive machines.

In all these contexts, among the major requirements for the EOS there will be, for example, the optimal allocation of logical resources while minimizing end-to-end network and application latencies. Let's see how this requirement has been taken into account in the EOS prototype design and development.

The EOS prototype leverages available open source software complemented with the development of other required software modules. In particular, the two main pieces of open source software are OpenStack and ONOS. The former will be used to manage the virtual machines executing the network and service functions of the virtual infrastructure; the latter will be in charge of managing the fabric of connections, while executing the control applications. EOS can be seen as an overarching operating system that runs on top of both OpenStack and ONOS.

The software architecture of the EOS prototype is shown in Fig. 5, where the circle represents code additions to OpenStack. These code additions mainly address the capability of OpenStack to handle chains of VMs (i.e., service chains) and the Nova-scheduler of OpenStack, which currently uses algorithms (i.e., Filter&Weight) for scheduling VMs in isolation, without considering the status of the underlying network links.

Looking at Fig. 5, from a purely architectural viewpoint, EOS looks similar to XOS [15]. On the other hand, there are major differences that should be highlighted.

XOS is a service orchestration layer that manages scalable services running in a central office

re-architected as a datacenter (CORD). On the other hand, EOS is a highly pervasive software architecture, i.e., it is extended up to the terminals (current and future ones), smart things, and elements of capillary networks (e.g., aggregations of sensors, actuators, etc.).

XOS unifies management of a collection of services that are traditionally characterized as being NFV, SDN, or cloud specific. EOS also addresses services that can also be executed at the edge (also leveraging edge and fog computing). This difference dictates profound implications, i.e., EOS is implemented with a scalable software architecture leveraging a trade-off between top-down and bottom-up intelligence.

In the specific use case shown in Fig. 6, a mobile cognitive machine is seen as an IE, with its own operative system. The control system of a mobile cognitive machine usually comprises many ROS nodes. For example, one node controls a laser range-finder, one node controls the wheel motors, one node performs localization, one node performs path planning, one node provides a graphical view of the system, one node for the cognitive service logics, and so on. Other remote ROS nodes may be required to provide other services. In this perspective, it can be argued that ROS nodes can be seen as service components of a service chain executed over the EOS.

Let's focus on a service request to allow a mobile cognitive machine to perform some articulated task (e.g., at the scene of a disaster) with ultra-low reacting times (e.g., a mobile robot being controlled remotely to act in an environment that is changing dynamically).

These requirements are dictating the need to execute said cognitive service by using a proper balance of local, edge, and centralized processing-storage resources. In fact, the machines' reaction times very much depend on IT response time and network latency, and even small changes in the area's layout, or delays in the actuation commands, can lead to catastrophic failures.

EOS, with the software architecture described previously, will be able to exploit this intelligence. For example, the selection method makes it possible to select the proper IEs to assign the execution of the service chain tasks, minimizing end-to-end network and application latencies.

Concluding Remarks

Broadband diffusion and ICT performance acceleration, coupled with cost reductions, are boosting innovation in several industrial and society sectors, thus creating the conditions for a socio-economic transformation, called softwarization. In particular, softwarization of telecommunications will make possible virtualizing network and service functions and executing them in software platforms fully decoupled from the physical infrastructure.

This article has focused attention on the edge of telecommunications infrastructures, arguing that softwarization will transform it in very powerful software platforms, enabling anything-as-a-service. EOS software architecture is proposed to achieve this, even in the short term. In fact, the development of EOS leverages available open source software. A use case was described to validate an EOS prototype with a proof-of-concept.

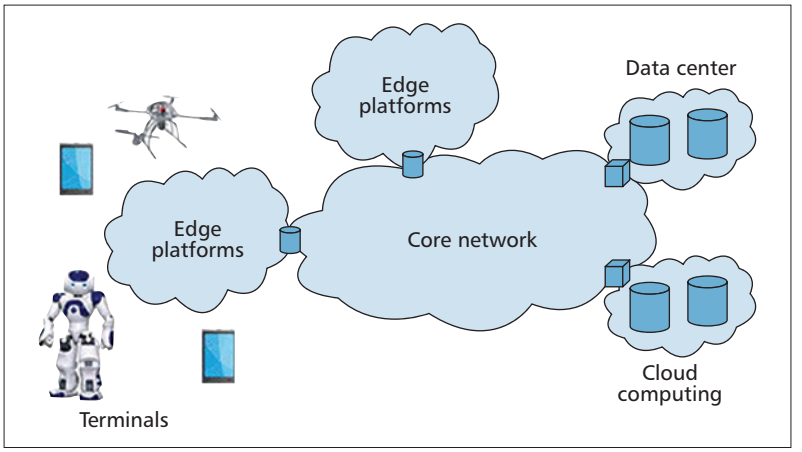

Figure 6. Use-case.

References

[1] "Software-Defined Networking: The New Norm for Networks," *Open Netw. Found.*, Palo Alto, CA, USA, white paper, available: https://www.opennetworking.org/.

[2] "Network Functions Virtualisation," *Eur. Telecommun. Std. Inst. (ETSI)*, Sophia-Antipolis Cedex, France, white paper, available: http://portal.etsi.org/NFV/NFV_White_Paper.pdf.

[3] "Mobile-Edge Computing," *Eur. Telecommun. Std. Inst. (ETSI)*, Sophia-Antipolis Cedex, France, white paper, available: https://portal.etsi.org/Portals/0/TBpages/MEC/Docs/Mobile-edge_Computing_-_Introductory_Technical_White_Paper_V1%2018-09-14.pdf.

[4] "OpenStack," available: http://www.openstack.org/.

[5] "ONOS," available: http://onosproject.org/.

[6] A. Manzalini *et al.*, "Software-Defined Networks for Future Networks and Services," presented at the *IEEE Workshop SDN4FNS*, Trento, Italy, Nov. 2013, white paper, available: http://sites.ieee.org/sdn4fns/whitepaper/.

[7] A. Manzalini and A. Stavdas, "The Network is the Robot," *Communications & Strategies, ERN Economics of Networks eJournal*, Dossier. No. 96 — 4th Quarter 2014.

[8] "Measuring Broadband America," FCC report, available: https://www.fcc.gov/measuring-broadband-america.

[9] "Robot Operating System," available: http://wiki.ros.org/ROS/Introduction.

[10] M. Sloman and E. Lupu, "Engineering Policy-Based Ubiquitous Systems," published by Oxford University Press on behalf of The British Computer Society, 2005; doi:10.1093/comjnl/bxh000.

[11] B. Martini, M. Gharbaoui, and P. Castoldi, "Cross-Functional Resource Orchestration in Optical Telco Clouds," *ICTON 2015*.

[12] R. Alfano and G. Di Caprio, "Turbo: An Autonomous Execution Environment with Scalability and Load Balancing Features," *IEEE Wksp. Distributed Intelligent Systems: Collective Intelligence and Its Applications*, 2006.

[13] "A YANG Data Model for System Management," Internet Engineering Task Force, Aug. 2014, available: https://tools.ietf.org/html/rfc7317.

[14] E. Brewer, "Towards Robust Distributed Systems," *Proc. 19th Ann. ACM Symp.Principles of Distributed Computing (PODC 00)*, ACM, 2000.

[15] "Central Office Re-architected as a Datacenter," available: https://wiki.onosproject.org/pages/viewpage.action?pageId=3441030.

Biographies

Antonio Manzalini (antonio.manzalini@telecomitalia.it) received the M. Sc. Degree in electronic engineering from the Politecnico of Turin. In 1990 he joined CSELT, which then became part of Telecom Italia. He is the author of a book on network synchronization (for SDH), and his RT&D achievements have been published in more than 100 papers. He was active in ITU as Chair of ITU-T Questions, leading standards developments on transport networks. Since 2000 he has had leading roles in several EURESCOM and European Projects. In 2008 he was awarded with the International Certification of Project Manager by PMI. In 2013 and 2015 he led two projects on software defined networks (SDN) funded by EIT – Digital. He is currently a senior manager in the Strategy & Innovation Dept. (Future Centre) of Telecom Italia. His main interests are innovative architectures and services enabled by SDN and NFV, primarily for 5G. He is General Chair of the IEEE initiative on SDN.

Noel Crespi holds masters degrees from the Universities of Orsay (Paris 11) and Kent (UK), a diplome d'ingénieur from Telecom ParisTech, and a Ph.D. and habilitation from Paris VI University (Paris-Sorbonne). In 1993 he joined CLIP, Bouygues Telecom and then went to Orange Labs in 1995. He took leading roles in the creation of new services with the successful conception and launch of Orange's prepaid service, and in standardization. In 1999 he joined Nortel Networks as a telephony program manager, architecting core network products for EMEA region. He joined Institut Mines-Telecom in 2002, and is currently a professor and program director, leading the Service Architecture Lab. He is also an adjunct professor at KAIST, an affiliate professor at Concordia University, and is the scientific director of the French-Korean laboratory ILLUMINE.