altice labs

Clouds bring sunshine to networks

Future networks; Cloudification; SDN; NFV; Softwarization

White paper

Version 1.0, April 2022

Introduction

Communication services providers (CSP) have long initiated the journey towards digitalization by evolving their information systems to cloud approaches, resulting in deeper business insights and improved customer experience. The network, however, is a different story: we are just starting to scratch the surface of what cloud technologies, enabled by virtualization and software-defined approaches, can do for telecom operations worldwide, together with a full ecosystem of service-based architectures, open innovation, and development and operations (DevOps). 5G is a good example of how networks are starting to move in this direction. In fact, not only the 5G architecture was designed considering a cloud-oriented implementation, but all modern networks and the services they support will have to move in that direction.

This is particularly evident with edge computing: as the public clouds evolve from a few high scale data centers towards processing at the edge to enable low-latency and high-bandwidth applications, like those related to augmented reality (AR), virtual reality (VR) or internet of things (IoT), and 5G access network aligns to provide the edge with the needed communications functionalities, the whole network edge is evolving to use the same hardware and the same practices that cloud providers use [1][2].

Network cloudification, hereby considered as the process leading to the adoption of cloud technologies on the network, is the culmination of a process of network softwarization [3] and builds on the achievements (and limitations) of network functions virtualization (NFV) and software-defined networks (SDN). The desirable result: software-oriented networks able to cope with fast change and permanent service innovation.

This trend presents many technical challenges, but above all, it involves deep transformation on how the CSP and solution providers operate, both internally and within the ecosystem. Namely, the extreme acceleration on the development/integration/delivery cycles brings new approaches like DevOps and continuous integration (CI)/ continuous delivery (CD) that create highly automated processes, not within reach of more traditional organizations. CSP seem to be gearing up, certainly pushed by the opportunity to swiftly evolve their service offerings and maintain their relevance in a world where just selling bandwidth becomes less and less profitable. **Figure 1,** using 2021 data, illustrates how CSP evaluate their own degree of preparedness:

This article starts by addressing the whole network softwarization process, its reasons, components, and main actors. Then, it delves deeper into the architectures and technologies supporting Network Cloudification, finalizing with the Altice Labs' approach to this matter, using a concrete example: the virtualized broadband network gateway (BNG).

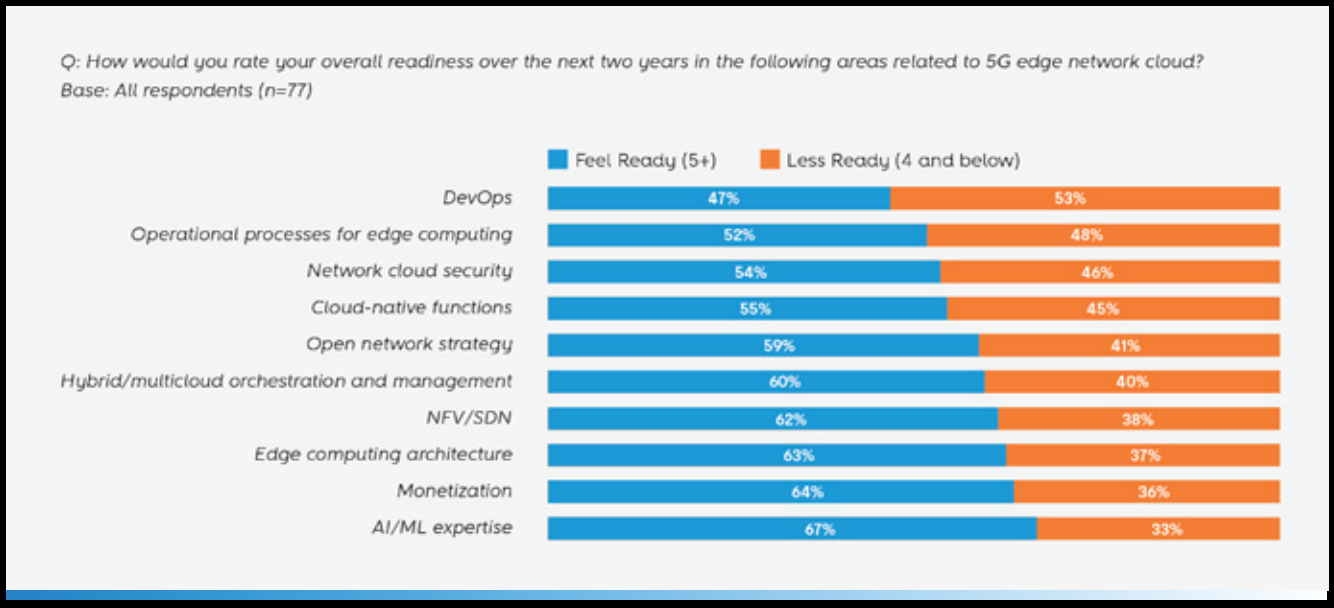


Figure 5 – Individual impact of the features on the model using SHAP values

Softwarization

The digitalization shift in the telecommunications world has led to significant pressure on the telecom operators to continuously increment their data networks capacity (as the number of connected devices explodes), bandwidth demand (as more data services are used simultaneously), and latency (as newer and more critical services need very close to real-time communication) to deliver faster and more reliable data services to their customers. The wide variety of data services and the rapid rate at which new ones are developed and commercialized today also means that telecom operators need to be able to quickly re-design parts of their network to cope with the specific requirements of some of those services.

To achieve such flexibility, mutability, and scalability, with a much bigger network capillarity, where network functions are evermore close to the end-user to provide more throughput and less latency, it is necessary to move away from the classical architecture – based on static deployment of vendor-specific hardware appliance characterized by monolithic functionality deployed at specific network locations – and adopt a much more dynamic architecture founded on what the International Telecommunication Union, Telecommunication Standardization Sector (ITU-T) defines as network softwarization: "an overall approach for designing, implementing, deploying, managing and maintaining network equipment and/or network components by software programming" [3]. Three main technologies/principles laid the ground for successful network softwarization: SDN, NFV, and cloud computing.

SDN [5] provides the foundations that allow operations to centrally define how a network function should treat users' traffic. It aims at making the network agile and flexible by introducing a software approach to the definition of how the hardware units – where the user traffic is processed – should behave across the entire network. These principles can be employed for managing a network function on the configuration/provision phases of the network setup and in real-time, where new processing rules can be applied depending on several conditions (e.g., type of traffic, source, destination, etc.). SDN approaches were boosted by another long-standing systems' architectural principle, the control and user plane separation (CUPS), where network systems are divided into two main parts: the user plane dealing with the user traffic and the control plane responsible for determining the behavior of user plane.

As per NFV [6], the virtualization of network functions allows new ways to design, deploy and manage networking services. NFV enunciates the process to decouple network functions from running on vendor and function-specific hardware, enabling the generalized use of commercial off-the-shelf (COTS) equipment for network functions. Virtualization allows the same computing environment (in its smallest instantiation set, it can be reduced to a single computing node) to run multiple independent systems at the same time, including all network traffic processing and the corresponding function's control software.

In the end, NFV makes it possible to instantiate a network function where and when it is needed. At the same time, SDN guarantees that traffic is correctly steered and the network function can be disaggregated and distributed as better suited.

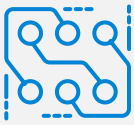
More recently, the widespread adoption of cloud computing technologies across all types of industries and organizations has forced telecom operators to look at this completely new paradigm of developing systems and conducting operations. On the one hand, systems by design should now intrinsically assure elasticity and scalability of their workloads to extract the most benefits of a distributed cloud ecosystem and assure a quick response to always evolving service needs. On the other hand, the cloud computing principles, applied to the operation side of system management, bring a more flexible, dynamic, and automated way to answer the challenges to continuously deliver the best quality of experience to each and every customer.

The top six benefits of Network Softwarization and cloudification can be summarized in:

Optimized allocation of computational resources

Centralized network control

Automated operations



Hardware and software vendor options

Network visibility

Fast time-to-market of new service

The push to more open and dynamic networks also allowed the flourishing of open initiatives that took advantage of the decoupling from vendor-specific hardware to create and incentivize the production of open-source software that, in turn, allows the participation of more players thus leading to an ever-growing more pulsating and diverse ecosystem. The Open Networking Foundation (ONF) is one popular initiative in this area, but others like The Linux Foundation, the Open Compute Project (OCP), or the Telecom Infra Project (TIP) are also very relevant projects.

Several standards developing organizations (SDO) have also taken the concept of network softwarization and cloudification into their definition of next-generation networks both on wired and wireless technologies. The most relevant SDO for mobile network and systems – 3rd Generation Partnership Project (3GPP) – has adopted from the "get-go" softwarization principles in the definition of its fifth generation (5G) architecture [7], clearly separating user plane from control plane functions and defining a service-based architecture that suits perfectly with the principles of cloud computing. Similarly, the Broadband Forum (BBF), the leading SDO for fixed broadband network and systems, has been working on the incorporation of these principles on its latest technical reports where a new virtualized and disaggregated network architecture is proposed [8], covering all the path from customer premises all the way up to the data center. Other relevant SDO such as the European Telecommunication Standards Institute (ETSI), the Internet Engineering Task Force (IETF), and the International Telecommunications Union (ITU) also produce work of extreme relevance for the success of the proposed technology and paradigm shifts.

Architectures and technologies supporting cloudification

CUPS paved the way into the initial stages of network function disaggregation, ensuring that basic principles from the software engineering domain, such as the single-responsibility principle (SRP), are respected and fulfilled. CUPS, however, only solves part of the scalability equation: existing networking software carries a heavy load of legacy code and dated software design patterns, even on the control plane side. Such artifacts are often called "monoliths", i.e., they follow a monolithic architecture in which functionally distinguishable aspects such as data input/output (I/O), processing, storage, and user interfaces are all tightly coupled and entangled together, instead of providing well-defined interfaces for component separation.

The solution to break the monolith is long known and has been extensively applied in the IT world, as illustrated in **Figure 2**.

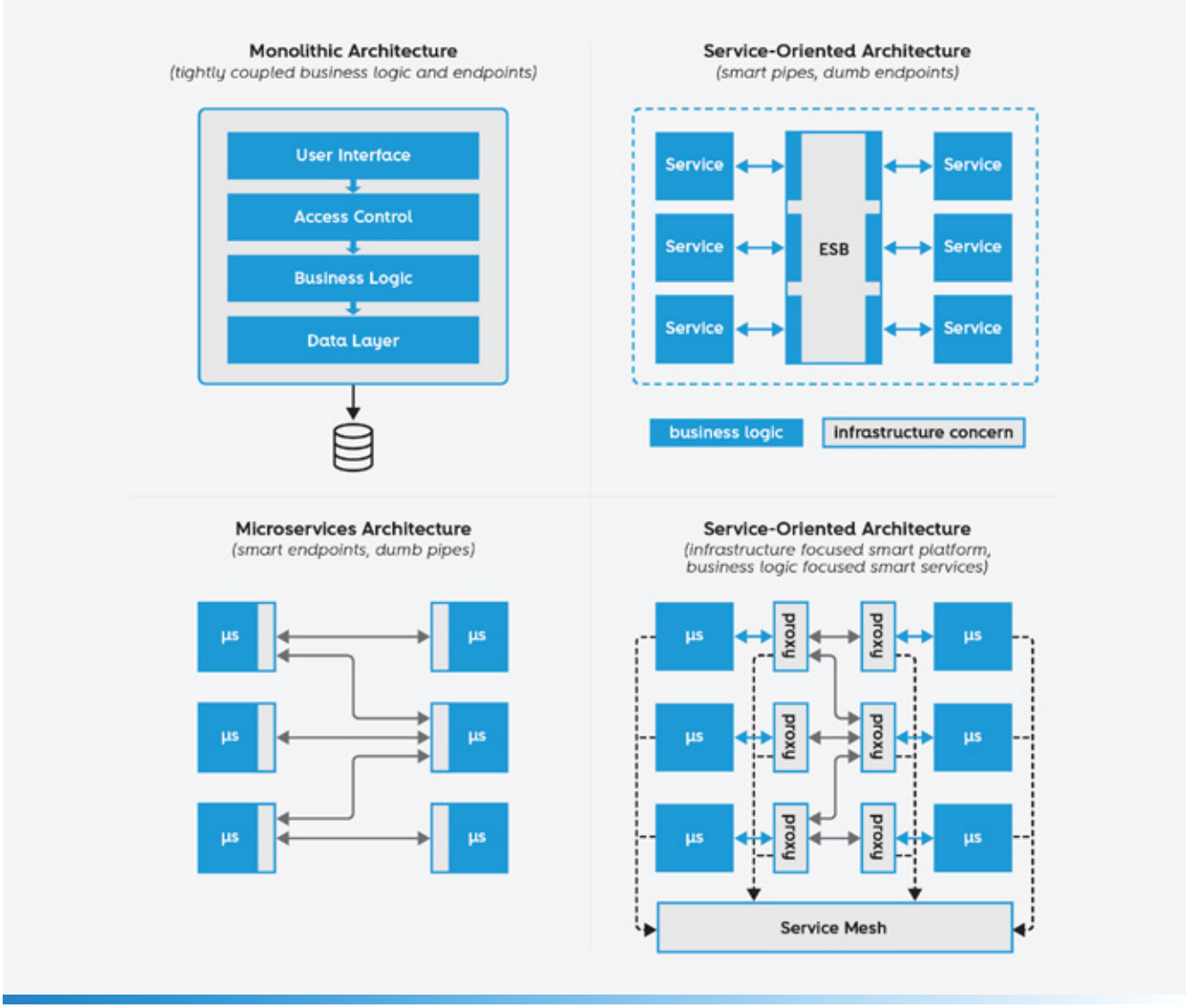


Figure 2 – The evolution of software architectures from the monolith cloud-native microservices

It involves a controlled and methodic process of refactoring the existing codebase by applying a sequence of small behavior-preserving transformations. Strictly focusing on the control plane, one of the most known design patterns to accomplish this transformation is the so-called service-oriented architecture (SoA). In SoA, complex software systems are broken into individual functional pieces (services) that expose an interface contract (typically a web service) and communicate with other services through a bus, often called the enterprise service bus (ESB). Service separation in SoA promotes the horizontal distributiveness of the overall software system, the interoperability between software vendors and allows the use of multiple redundant copies of the same service as a fallback for high availability [9].

Unlike classical distributed systems, where traffic distribution relied on the extensive use of load-balancers with statically provisioned service location, SoA tackles the problem from a different angle, with the introduction of a central discovery and registration entity (often a broker) which all the services use to register, authenticate, discover other services and send/receive data. If we look at modern network core architectures like 5G [6], it is no surprise to see that 3GPP chose to use a service-based paradigm in addition to the purely functional, interface-based approach. There, in addition to other core services, the network function repository function (NRF) provides a single record of all network functions available in a given mobile network, together with its profile and the services they support. Essentially, it allows network functions (NF) to discover other NF and to subscribe and get notified about the registration of new NF instances of a given type, having deep similarities with the broker role described above.

Despite the singular advantages of SoA, it should only be seen as an intermediary solution in the long network cloudification journey. SoA helps to decouple at a macro level but does not address the issues of granular scalability at the service level, i.e., it does not consider that the network function itself might be composed by a set of different components (a monolith in itself) nor the replication of these network functions across multiple datacenters. Additionally, SoA does not offer a clear solution to the load balancing issue.

This is of utmost importance in today's cloud-native world where, due to the advent of container-based virtualization, microservices are becoming the de-facto way to develop, ship, and distribute applicational workloads. The rise of the cloud-native application and the popularization of container orchestration frameworks, like Kubernetes, started to give visibility to complementary design patterns such as the service mesh architecture. The service mesh adds a layer of reliability, security (like mutual transport layer security (TLS) authentication), and observability features to a microservice application by inserting those features at the platform layer rather than in the application itself. Secure service-to-service communication is thus seen as a vital part of the application runtime behavior [10].

The service mesh architecture deeply inherits the CUPS principles. It is typically implemented as a set of scalable network proxies which run alongside the application code – sometimes referred to as "sidecars". The proxies handle the communication between microservices and are automatically injected by the platform. As such, they act as a point where the service mesh observability, security and tracing features are introduced with a minimal operational burden. Stateless in nature, sidecars can be seen as the user plane of the service mesh as they touch every request in the system without being aware of any configuration settings or policing. Features such as routing rules, request replication, load balancing settings, retrials, circuit breaking, etc., are inserted by a common control plane entity that ends up transforming all the individual data-planes into a transparent distributed

system. The service mesh pattern might sound complex to understand and involve a lot of magic, but, in the end, it all comes down to the exceptional event-driven nature of modern container orchestration frameworks.

Let's take a closer look at the 3GPP 5G specification again. We can see the service mesh is now also a fundamental part of the architecture: in Release 16 (TS23.501) [6], the service communication proxy (SCP) was introduced to support the 5G service-based architecture (SBA) in distributed multi-access edge deployments, effectively adding support for services meshes within the 5G SBA (**Figure 3**).

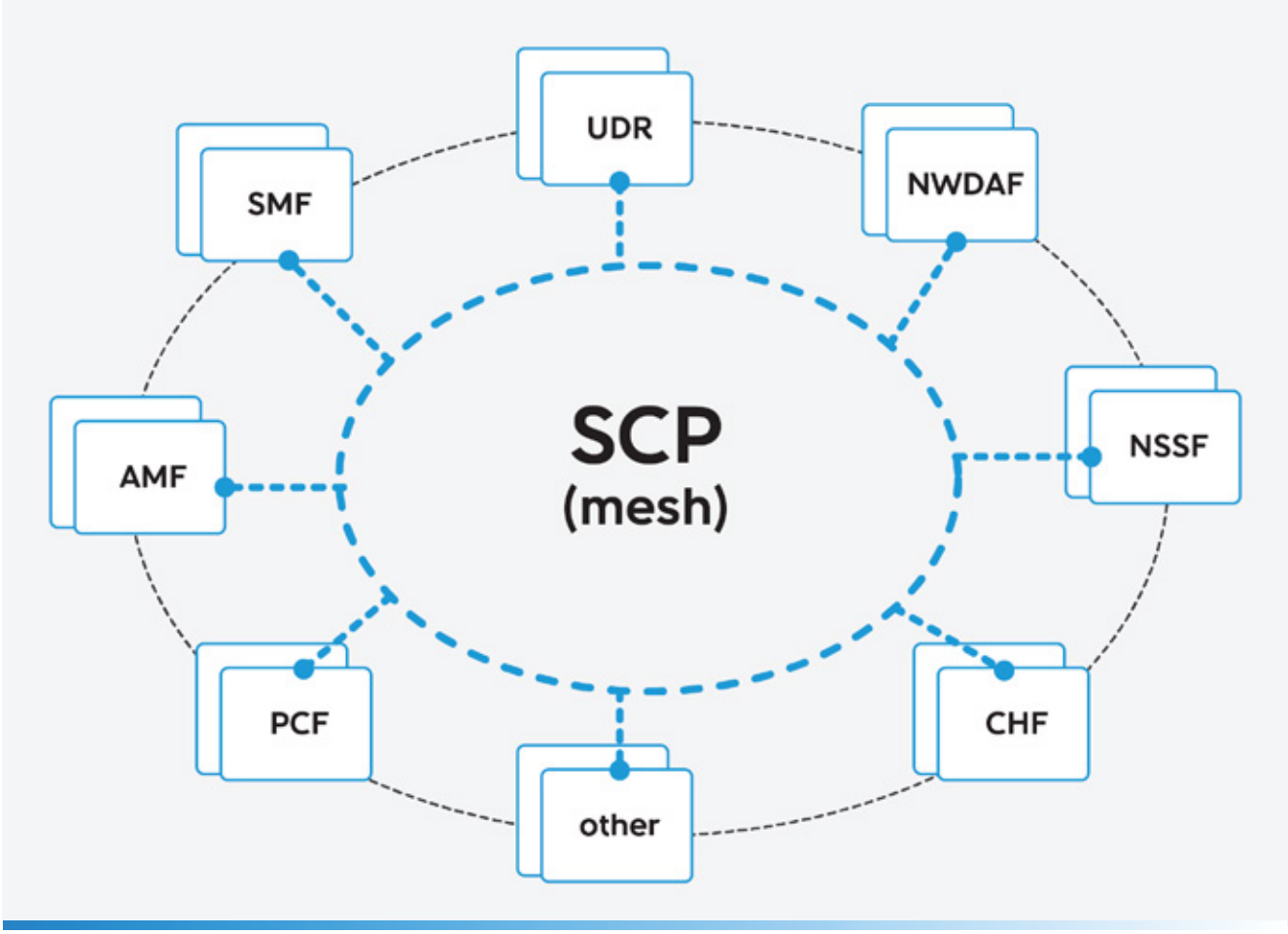


Figure 3 – 5G core service delivery method through SCP

While the control plane is following a clear architectural trend, favoring a cloud-native approach, microservice-based, by exploiting known design patterns such as the service-oriented architecture or the service mesh, the user plane is focused on a different challenge: the need for speed. Adopting the same principles in the user plane is severely affected by the caveats of network packet processing in virtualized environments.

In the (now) early days of I/O virtualization, network interface card (NIC) virtualization options were solely based on software. Between the virtual NIC associated with a virtual machine (VM) and the physical NIC of the server, the virtual machine monitor (VMM) was always the main active element in the virtualization process. Just like other I/O subsystems, network packet processing is interrupt-driven. When a packet is received by the server physical NIC, an interrupt request (IRQ) is sent to the server CPU, which must stop the currently running job to retrieve the data from the NIC buffers. Combining this with a virtualization scenario (e.g., a virtual machine) makes the situation even worse: the CPU core running the VMM needs to be interrupted but, after figuring out to which VM the packet must be sent, another interrupt must be generated so the CPU core(s) allocated to the VM can obtain the data [11]. The whole process is extremely slow and has a detrimental effect on the overall performance since the CPU constantly gets busy with non-strictly process-related tasks. In response to these problems, manufacturers have developed and standardized new approaches.

The single root I/O virtualization and sharing specification (SR-IOV) was released in 2007 by the Peripheral Component Interconnect Special Interest Group (PCI-SIG) as an extension to the PCI express (PCIe) specification to standardize the input-output memory management unit (IOMMU). With SR-IOV, a device such as a network adapter can separate the access to its resources among different PCIe hardware functions, splitting them into a PCIe physical function (PF) and one or more PCIe virtual functions (VF). VF can share physical resources such as memory and a network port with the PF and other VF [12]. In summary, SR-IOV allows network traffic to effectively bypass the VMM, moving it directly to the appropriate VF partition linked to a specific virtual machine context – a process known as direct memory access (DMA). **Figure 4** below illustrates SR-IOV.

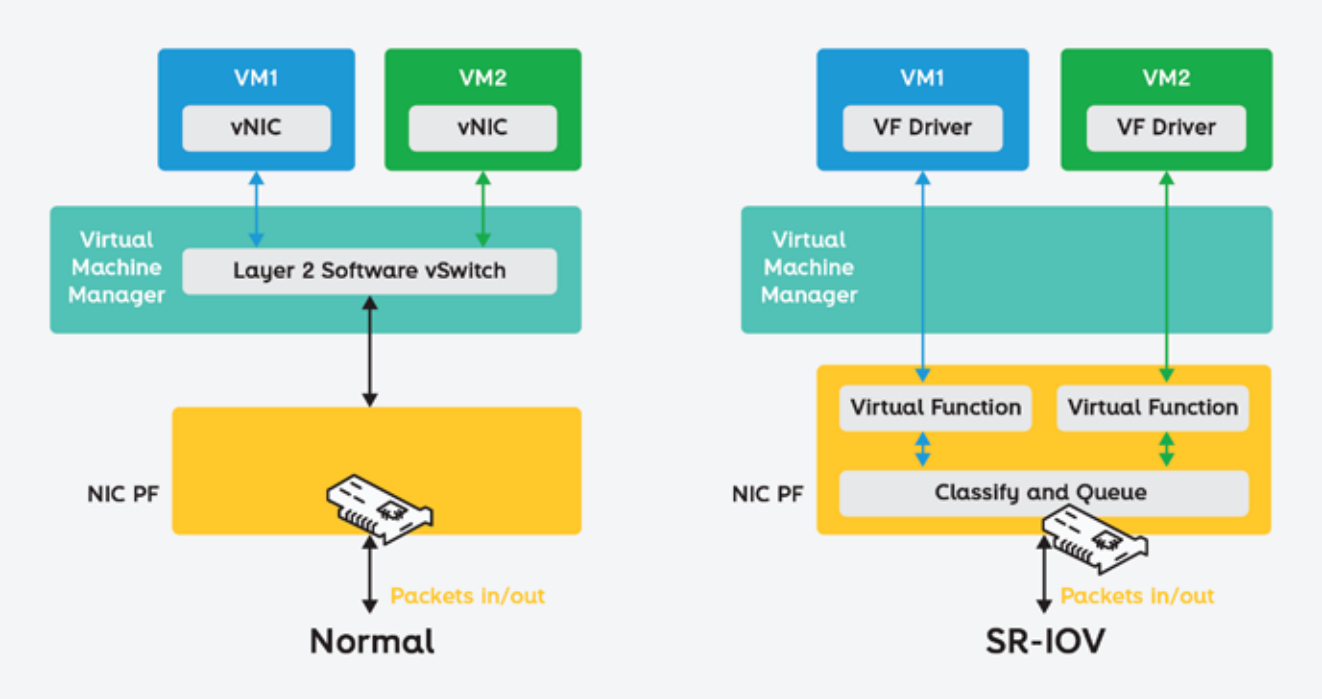


Figure 4 – SR-IOV bypassing the VMM layer

SR-IOV solves part of the performance drawbacks of data plane virtualization. Still, it only covers the path between the physical NIC and the VM. It does not address further performance bottlenecks that may reside in the operating system (OS) of the VM itself. Modern operating systems, like Linux distributions, segregate memory and hardware privilege access into two main hierarchical domains: the kernel space (the uppermost trusted resource) and the user space (where the application workload runs). For applications to send or receive network packets, interrupts need to be generated, data copied between both memory regions, and system calls executed. System calls involve context switching: the context is switched from kernel mode back to user mode consuming a significant amount of resources. Progress was made on the kernel side with the introduction of the new programming interface (NAPI), which is able to combine interrupts with requests: the NIC first works in interrupt mode but, under periods of high traffic, the interrupt is disabled, and the system periodically polls a queue for new packets. Nevertheless, without relying on other strategies, the performance of Linux user-space network applications is still far from ideal [13].

This fact has led some networking software to extend the user plane logic to the kernel space, with the development of support Linux kernel modules (LKM) loaded at runtime and without the need of recompiling the kernel or rebooting the machine. A noticeable example is the default operation mode of OpenvSwitch (OVS), a known virtual switch implementation. OVS inserts an LKM that maintains an exact-match flow cache of recently forwarded packets, updated from user space via an inter-process communication (IPC) mechanism. Since only the traffic that does not have a corresponding entry in the cache needs to make the IPC context switch, OVS is said to have production-grade forwarding performance. Note, however, that LKM may introduce security risks (custom code with kernel privileges) and are also not portable (they depend on the kernel version).

To overcome the above limitations, the extended Berkeley packet filter (eBPF) recently arose as a way to execute packet filtering logic as bytecode at the kernel space. Unlike LKM, eBPF bytecode is just-in-time compiled and verified by a set of tools that reside within the kernel. Ultimately, it runs on a restrictive and self-contained in-kernel VM, providing a safe environment for applications to package kernel-level code and interact with protected hardware resources while still having shared memory constructs accessible from user space [14]. The express data path (XDP), which combines eBPF benefits, was later introduced as a means for user-space applications to directly read RX packet pages out of a kernel driver without allocating any buffers or software queues. A direct access facility that still accounts for the presence of (and is provided by) the network stack of the kernel itself [15].

The use of kernel facilities like eBPF and XDP in virtualized network applications is still in its early days. Alternative "non-kernel" user plane acceleration approaches, like the data plane development kit (DPDK), are rather mature and have already been proven successful by a wide range of applications. Just like how SR-IOV bypasses COTS servers' VMM layer, we can say DPDK does the same for the Linux kernel. DPDK is a set of libraries that can be used to implement user-space drivers for NIC and fully realize the data plane in software with control over traffic queueing, memory, and buffer management. The framework can implement zero-copy DMA into large ring buffers (using memory hugepages) and, at the same time, it can rely on a polling mode for packet acquisition, just like NAPI [13]. **Figure 5** shows the various possibilities for data plane acceleration under Linux.

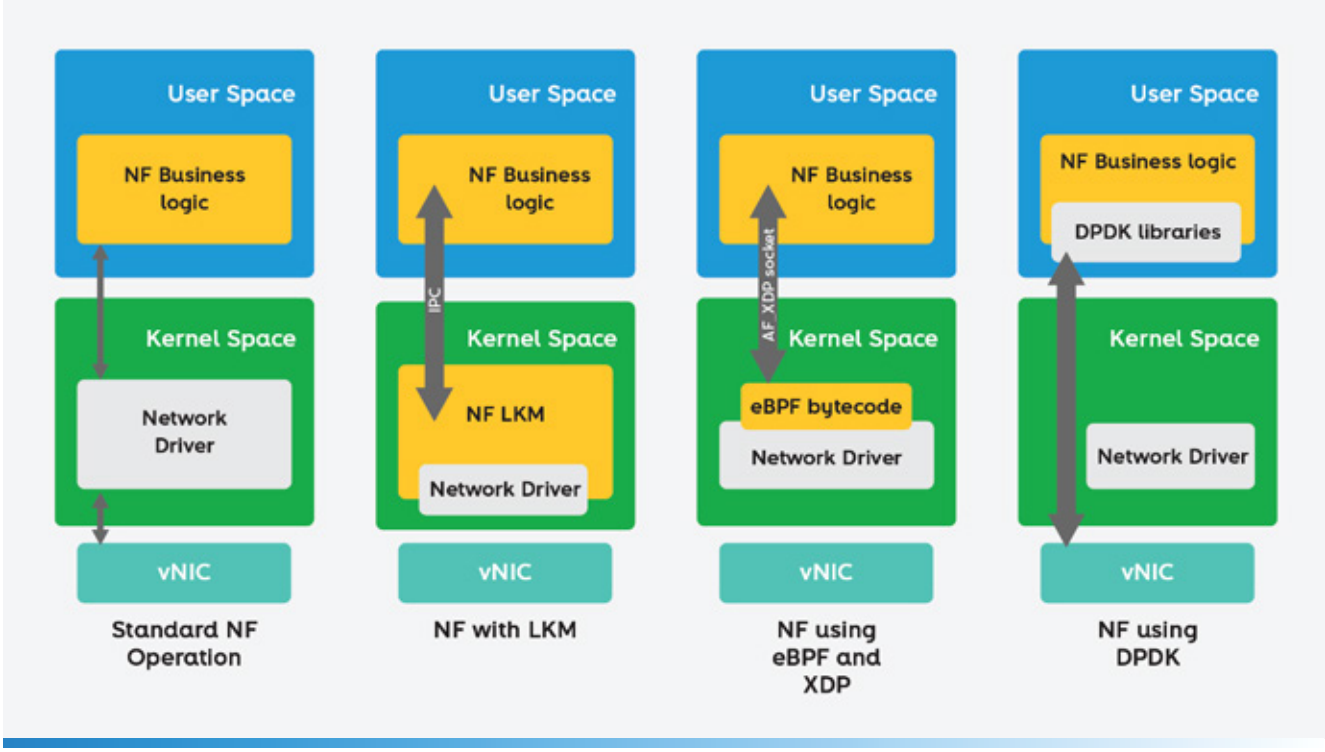


Figure 5 – Data plane acceleration options in Linux-based systems

As it bypasses the kernel networking layer, DPDK encouraged the development of purely software-based network stacks. Vector packet processing (VPP) is likely the most prominent example: a graph-based processing pipeline where custom functionalities (plugins) can be developed as new nodes in the overall graph [16]. Other projects like OVS also provide an alternative data path for DPDK enablement. It is important to refer that all those user plane acceleration facilities - even DPDK - are compatible with the microservices approach and modern container runtime orchestration frameworks. Kubernetes, for example, has adopted the container network interface (CNI) specification for managing network resources on a cluster with plenty of available plugins.

As seen, DPDK (and NFV in general) can place a significant strain on the processing capacity of the server CPU. When multiple gigabits of data need to be processed per unit of time, it might be desirable to offload some of the networking processing functions to dedicated hardware. In this field, a worth mentioning is deserved to smart NIC and programmable hardware switches, which employ specialized processors, field-programmable gate arrays (FPGA), to power their offload capacity [17]. Their FPGA can be programmed with standard development tools by taking advantage of high-level user-plane pipeline programming languages like P4 [18].

Features such as load balancing, encryption, packet encapsulation, deep packet inspection, or other intensive I/O tasks can be realized in hardware rather than software, saving the precious CPU cycles only for the control plane workloads.

All things considered, and as Frederick Brooks mentions in his software engineering classic "The Mythical Man-Month ", we can say there is no "silver bullet" when it comes to virtualizing and cloudifying network functions. Multiple paths are available, and options should be carefully chosen in light of the expected scale, throughput, and other design requirements. Nevertheless, one thing we know for sure: the days of the network function monolith are now long gone.

Altice Labs' approach

Altice Labs is pursuing its own strategy for network softwarization/cloudification, first of all by adapting its portfolio to cloud-based solutions. Also, to support a structured and systematic approach, Altice Labs has defined its own view for the access and edge networks, aligned with the ongoing standardization effort and best practices.

The diagram in **Figure 6**, available on next page, illustrates the high-level architecture defined by Altice Labs for a softwarized network edge which is thoroughly described in a former article [19]. Here, aspects like a virtualized infrastructure, CUPS, and function disaggregation are evident and define a framework for Altice Labs' network portfolio evolution.

Based in this architecture, corresponding to the highlighted area in **Figure 6**, and already capitalizing on the benefits described above, new products are being developed. One example is the virtualized BNG (vBNG) case, in which the disaggregated and cloudified approach can give rise to a next-generation product and also create a foundation of functional components that will be available for many other products and solutions.

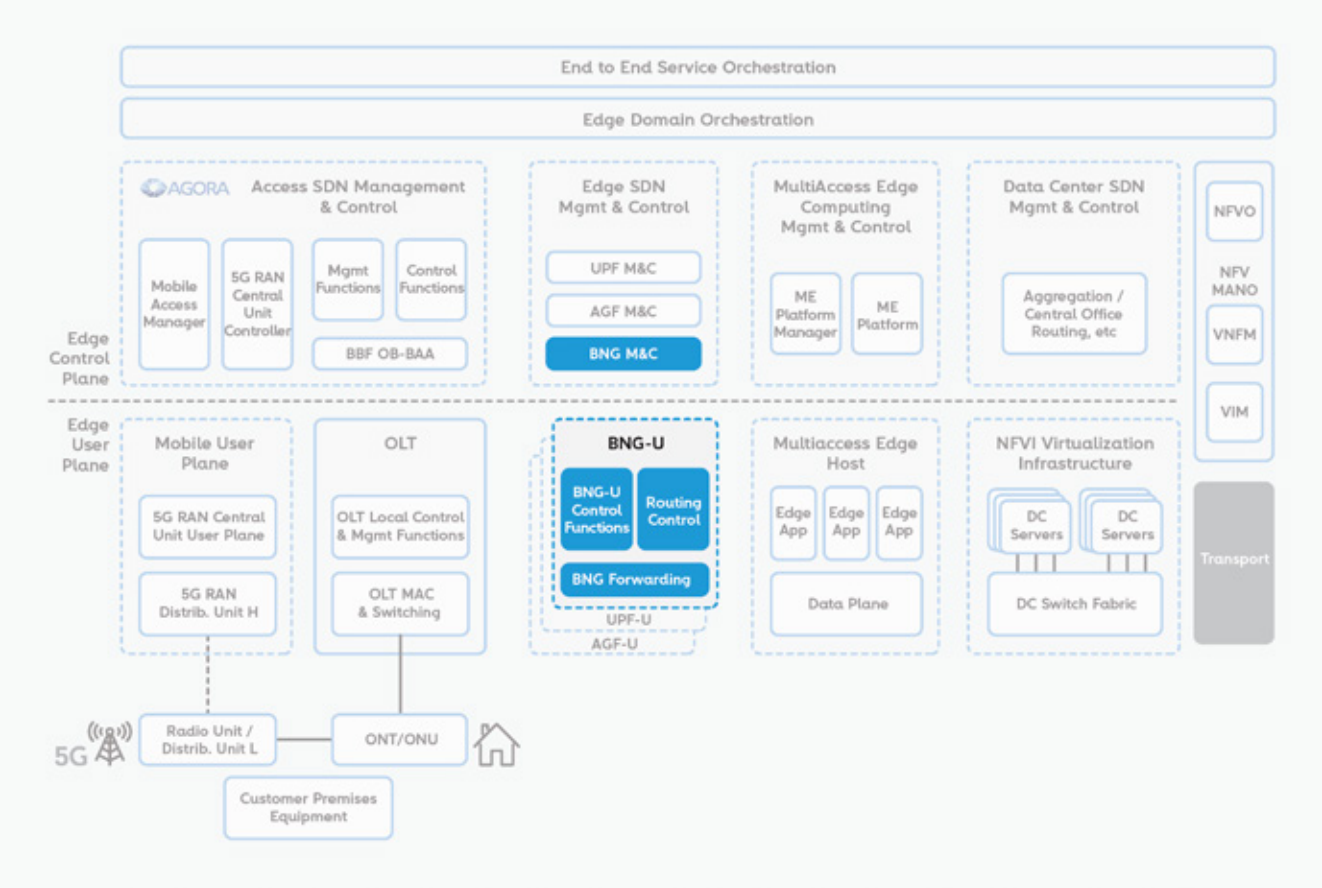


Figure 6 – Altice Labs' edge architecture

The BNG plays a central part in nowadays fixed broadband networks as it is responsible for the interconnection of the access network and service networks (e.g., private service networks, telecom operator's service networks, internet). Often, its functional scope also includes assuring layer 2/3 connectivity between access and service networks, IP traffic routing, customer IP assignment and management, QoS enforcement, session accounting, lawful interception, multicast traffic (for IPTV), and network address translation (NAT).

Traditionally, as with all other network functions, the BNG is delivered as a vendor-specific appliance (running on vendor-specific hardware). As already discussed in this article, this type of approach presents several shortcomings for successfully tackling the challenges raised by nowadays demand for very fast response to new service requirements, which may imply network architectural changes.

In Altice Labs' point of view, the natural path for the BNG is the adoption of the network softwarization principles (as it is currently being defined by BBF in its TR-459 series) [20] and its development and delivery following cloud-native software principles.

The first step for the development of a vBNG was the functional separation between the control plane and the user plane. **Figure 7** shows that separation in three levels.

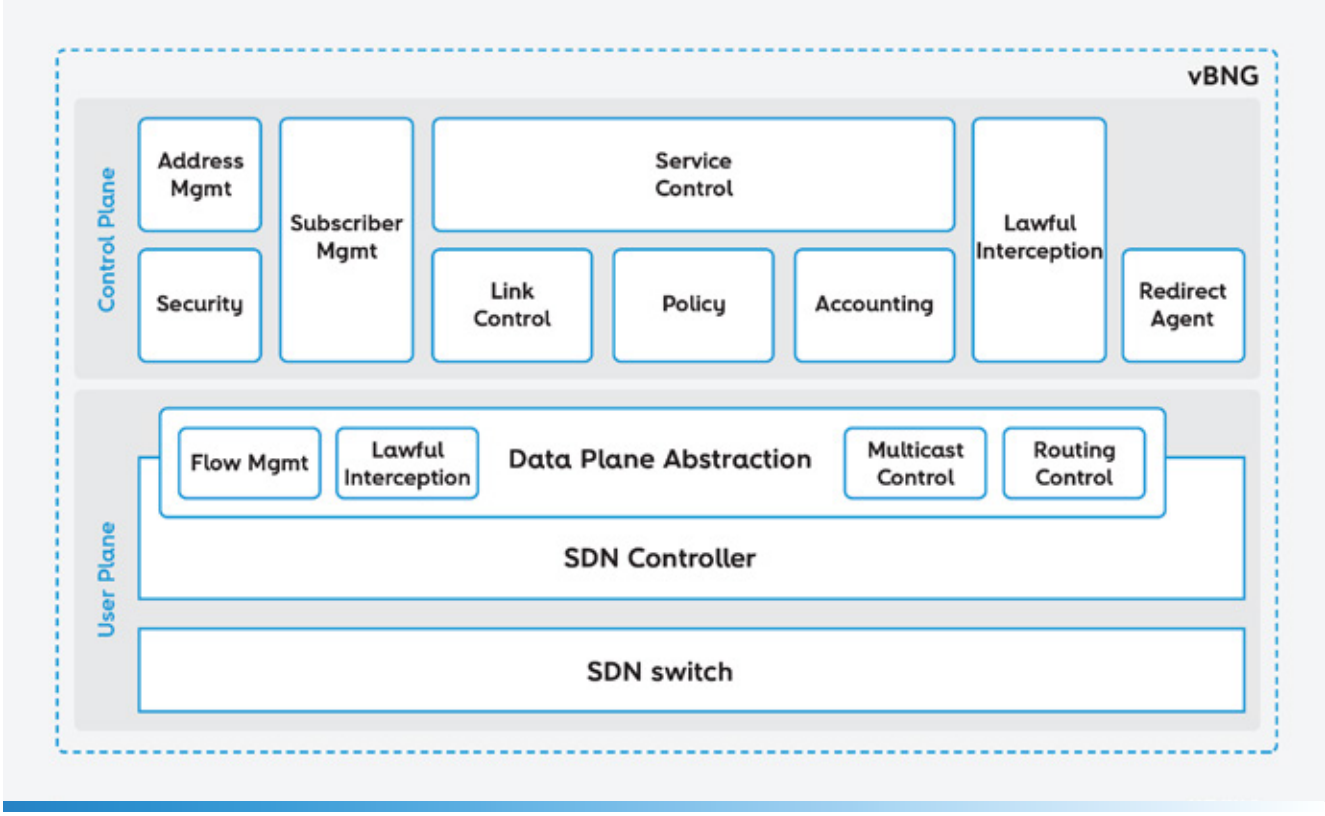
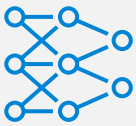


Figure 7 – Altice Labs' vBNG architecture

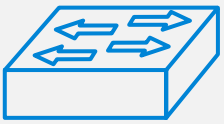
Control Plane Layer (CP)

It is where all the control services for the different functionalities reside and is responsible for determining the system's behavior as a whole.



Data plane abstraction layer (DPA)

It is responsible for the close control of the SDN switch and behaves as a middleware between the switch and the control plane functions and, together with the SDN switch layer, forms the user plane of the system.



SDN switch layer

It is responsible for switching and forwarding customer data packets.

In this architecture, the control plane is divided into services (using a micro-services approach) that are independent and are responsible for a specific set of functionalities. They are deployed using containers and managed by a Kubernetes cluster. This way, each service can be instantiated and adapted individually to respond to a specific demand, largely reducing the impacts on the system. The adoption of a cloud-native approach allows for zero-touch and minimal downtime when introducing new or augmented services that require changes in the system and/or the network.

The user plane (DPA and SDN switch) is also deployed using containers managed by a Kubernetes cluster providing the desired degree of flexibility to easily create and deploy new network configurations. Currently, the SDN switch used in the vBNG is the OVS, which has the advantage of being instantiated on COTS servers with x86 architecture, thus allowing computational resource sharing with other vBNG components and/or network functions. Additionally, the OVS has inherently available some packet acceleration options that allow its use in scenarios demanding very large throughputs. Other approaches, based on OCP compliant SDN switches, are currently under analysis as an alternative to the OVS to address higher performance requirements. Regarding the DPA, this layer includes an SDN controller that controls the underlying SDN switch and exposes a set of API (configuration, control, and packet redirection) to the CP. This approach assures an independence of the CP from the selected SDN controller as the DPA is able to create an abstraction on the specifics of each SDN controller northbound API.

The adoption of CI/CD methodologies on the development cycle allows the implementation of a much more agile process of releasing and delivering new versions, minimizing service impacts, and guaranteeing service continuity, which is paramount as changes become more frequent due to all the factors explained in previous sections.

Finally, the architecture designed for the vBNG provides the necessary tools to introduce a much more efficient way to operate the system as it allows several deployment architectures with logically centralized control plane and distributed user plane.

Conclusion

SDN switch layerWith the maturity of the technologies described above and the vibrant ecosystem that is now extending from CSP and traditional solution providers to Internet hyperscalers and open-source communities, networks are certainly up for deep transformations. The relation built on many years of mutually beneficial collaboration between CSP and large telecom technology vendors is taking its time to adapt, but the rationale, the tools, and new players for a cloudified network are here, and some are already starting to use them to their advantage [21].

References

[1] Open Networking Foundation, “CORD”, 2021 [Online]. Available: https://opennetworking.org/cord/

[2] O-RAN Alliance, “O-RAN Alliance”, 2021 [Online]. Available: https://www.o-ran.org/

[3] ITU-T, Y.3150, “High-level technical characteristics of Network Softwarization for IMT-2020”, September 2020

[4] IBM and 451 Research, “Network Cloudification , 5G and Edge Computing - Strategic Perceptions and Practitioner Views,” pp. 1–16, 2021, [Online]. Available: https://www.ibm.com/search?lang=en&cc=us&q=Network%20Cloudification%20%2C%205G%20and%20Edge%20Computing%20-%20Strategic%20Perceptions%20and%20Practitioner%20Views

[5] Open Networking Foundation, TR-521, “SDN Architecture 1.1”, February 2016

[6] ETSI, “Network Functions Virtualization (NFV)”, 2021 [Online]. Available: https://www.etsi.org/technologies/nfv

[7] 3GPP, TS 23.501, ” System Architecture for the 5G System; Stage 2; V16.10.0”, September 2021

[8] Broadband Forum, TR-384, “Cloud Central Office Reference Architectural Framework”, Issue 1, January 2018

[9] Red-Hat, “What is service-oriented architecture (SOA)?”, 2021 [Online]. Available: https://www.redhat.com/en/topics/cloud-native-apps/what-is-service-oriented-architecture

[10] Metaswitch, “The Service Communication proxy 5g caught up in a service mesh”, 2021, [Online]. Available: https://www.metaswitch.com/blog/the-service-communication-proxy-5g-caught-up-in-a-service-mesh

[11] MetaSwitch, “Accelerating the NFV data plane”, 2021 [Online]. Available: https://www.metaswitch.com/blog/accelerating-the-nfv-data-plane

[12] Microsoft, “Overview of single root i/o virtualization”, 2021 [Online]. Available: https://docs.microsoft.com/en-us/windows-hardware/drivers/network/overview-of-single-root-i-o-virtualization--sr-iov-

[13] Selectel, “Introduction to DPDK Architecture Principles”, 2021 [Online]. Available: https://blog.selectel.com/introduction-dpdk-architecture-principles/

[14] ebpf.io, “eBPF”, 2021 [Online]. Available: https://ebpf.io/

[15] Linux Networking Documentation, “AF_XDP”, 2021 [Online]. Available: https://01.org/linuxgraphics/gfx-docs/drm/networking/af_xdp.html

[16] Fd.io, “The universal dataplane”, 2021, [Online]. Available: https://fd.io/

[17] TechTarget, “An introduction to smart NICs and their benefits”, 2019 [Online]. Available: https://www.techtarget.com/searchnetworking/tip/An-introduction-to-smart-NICs-and-their-benefits

[18] P4 Working Group, “The P4 Open-Source Programming Language”, 2021, [Online]. Available: https://p4.org/

[19] A. Brízido, G. Gaspar, M. Santos, R. Calé and V. Mirones, “Redesigning the network edge for a new era,” InnovAction #5, vol. 1, no. 5, pp. 103–115, 2020

[20] Broadband Forum, TR-459, “Control and User Plane Separation for a disaggregated BNG”, Issue 1, June 2020

[21] Joanne Taaffe, TM Forum Inform, “Networks follow OSS/BSS into the public cloud”, 2021 [Online]. Available: https://inform.tmforum.org/insights/2021/06/networks-follow-oss-bss-into-the-cloud/

Acronyms

3GPP	Third Generation Partnership Project, a collaboration between groups of telecommunications standards associations
5G	Fifth generation mobile networks
AF_XDP	Na address family, optimized for high performance packet processing and presented into the Linux Kernel 4.18
AGF	Access Gateway Function
AMF	Access and Mobility Function
API	Application Programming Interface
AR	Augmented Reality
BBF	Broadband Forum
BNG	Broadband Network Gateway
CHF	Charging Function
CI/CD	Continuous Integration/Continuous Delivery
CNI	Container Network Interface
COTS	Commercial Off-The-Shelf
CPU	Central Processing Unit
CSP	Communications Service Providers
CUPS	Control and User Plane Separation
DC	Data Centre
DevOps	Software development methodology that combines software development with information technology operations
DMA	Direct Memory Access
DPA	Data Plane Abstraction
DPDK	Data Plane Development Kit
eBPF	extended Berkeley Packet Filter
ESB	Enterprise Service Bus
ETSI	European Telecommunications Standards Institute
FPGA	Field Programmable Gate Array
I/O	Input/Output
IETF	Internet Engineering Task Force, an open standards organization that develops and promotes voluntary Internet standards
IOMMU	Input-Output Memory Management Unit

IoT	Internet of Things
IP	Internet Protocol
IPC	Inter-Process Communication
IPTV	Internet Protocol Television
IRQ	Interrupt Request
IT	Information Technologies
ITU-T	International Telecommunication Union, Telecommunication Standardization Sector
LKM	Linux Kernel Modules
MANO	Management and Network Orchestration
NAPI	New API
NAT	Network Address Translation
NF	Network Function
NFV	Network Function Virtualization
NFVI	NFV Infrastructure
NFVO	NFV Orchestrator
NIC	Network Interface Card
NRF	Network Function Repository Function
NSSF	Network Slice Selection Function
NWDAF	Network Data Analytics Function
OB-BAA	Open Broadband - Broadband Access Abstraction
OCP	Open Compute Project, an organization that shares designs of data centre products among companies
OLT	Optical Line Termination
ONF	Open Networking Foundation, a user-driven organization dedicated to the promotion and adoption of SDN through open standards development
ONT	Optical Network Terminal
ONU	Optical Network Unit
OS	Operating System
OVS	Open vSwitch, an open-source implementation of a distributed virtual multilayer switch
PCF	Policy and Charging Function
PCIe	PCI express

PCI-SIG	Peripheral Component Interconnect Special Interest Group, an electronics industry consortium
PF	Physical Function
QoS	Quality of Service
RAN	Radio Access Network
RX	Reception
SBA	Service-Based Architecture
SCP	Service Communication Proxy
SDN	Software-Defined Network
SDO	Standards Development Organization
SMF	Session Management Function
SoA	Service-Oriented Architecture
SR-IOV	Single Root I/O Virtualization
SRP	Single-Responsibility Principle
TIP	Telecom Infra Project
TLS	Transport Layer Security
UDR	Unified Data Repository
vBNG	Virtual BNG
VF	Virtual Functions
VIM	Virtual Infrastructure Manager
VM	Virtual Machine
VMM	Virtual Machine Monitor
VNF	Virtual Network Function
VNFM	VNF Manager
vNIC	Virtual Network Interface Card
VPP	Vector Packet Processing
VR	Virtual Reality
XDP	eXpress Data Path

Authors

Rui Calé

Senior Consultant and Architect

Altice Labs, Portugal

cale@alticelabs.com

www.linkedin.com/in/rui-cal%C3%A9-0306132/

Miguel Borges de Freitas

Technology Development Manager

Altice Labs, Portugal

miguel-r-freitas@alticelabs.com

Miguel Santos

Product Management

Altice Labs, Portugal

miguel-c-santos@alticelabs.com

Contacts

Address
Rua Eng. José Ferreira Pinto Basto
3810 - 106 Aveiro (PORTUGAL)

Phone
+351 234 403 200

Media
contact@alticelabs.com
www.alticelabs.com