A Cloud Optical Access Network for Virtualized GNSS Receivers

C. Fernández–Prades, *Centre Tecnològic de Telecomunicacions de Catalunya (CTTC/CERCA)*, Spain.

C. Pomar, *T-Systems Iberia*, Spain.

J. Arribas, *Centre Tecnològic de Telecomunicacions de Catalunya (CTTC/CERCA)*, Spain.

J.M. Fabrega, *Centre Tecnològic de Telecomunicacions de Catalunya (CTTC/CERCA)*, Spain.

J. Vilà-Valls, *Centre Tecnològic de Telecomunicacions de Catalunya (CTTC/CERCA)*, Spain.

M. Svaluto Moreolo, *Centre Tecnològic de Telecomunicacions de Catalunya (CTTC/CERCA)*, Spain.

R. Casellas, *Centre Tecnològic de Telecomunicacions de Catalunya (CTTC/CERCA)*, Spain.

R. Martínez, *Centre Tecnològic de Telecomunicacions de Catalunya (CTTC/CERCA),* Spain.

M. Navarro, *Centre Tecnològic de Telecomunicacions de Catalunya (CTTC/CERCA)*, Spain.

F. J. Vílchez, *Centre Tecnològic de Telecomunicacions de Catalunya (CTTC/CERCA)*, Spain.

R. Muñoz, *Centre Tecnològic de Telecomunicacions de Catalunya (CTTC/CERCA)*, Spain.

R. Vilalta, *Centre Tecnològic de Telecomunicacions de Catalunya (CTTC/CERCA)*, Spain.

L. Nadal, *Centre Tecnològic de Telecomunicacions de Catalunya (CTTC/CERCA)*, Spain.

A. Mayoral, *Centre Tecnològic de Telecomunicacions de Catalunya (CTTC/CERCA)*, Spain.

BIOGRAPHIES

Dr. Carles Fernández–Prades holds the position of Senior Researcher and serves as Head of the Statistical Inference for Communications and Positioning (SI) Department at the Centre Tecnològic de Telecomunicacions de Catalunya (CTTC/CERCA), where he previously served as Head of the Communication Subsystems Area (2006-2013) and Head of the Communications Systems Division (2013-2016). He received a PhD degree in Electrical Engineering from Universitat Politècnica de Catalunya (UPC) in 2006.

Mr. Christian Pomar works for T-Systems Iberia, the Spanish subsidiary of Deutsche Telekom, as Head of Systems Integration Sales. He received the MSc degree in Telecommunication Engineering from UPC in 1996. His primary areas of interest include Internet of Things, Virtual Reality and Supercomputing and their applications.

Dr. Javier Arribas holds the position of Senior Researcher in the SI Department at CTTC/CERCA. He received the PhD degree in Signal Processing from UPC in 2012 and the BSc and MSc degrees in Telecommunication Engineering in 2002 and 2004, respectively, at La Salle University in Barcelona, Spain.

Dr. Josep M. Fabrega holds the position of Senior Researcher in the ONS Department at CTTC/CERCA. He received the BSc and MSc degrees in Telecommunication Engineering and the PhD degree in Signal Theory and Communications from UPC in 2002, 2006 and 2010, respectively.

Dr. Jordi Vilà-Valls holds the position of Researcher in the SI Department at CTTC/CERCA. He received the PhD degree in Electrical Engineering from Grenoble INP (INPG), France, in 2010.

Dr. Michela Svaluto Moreolo is a Senior Researcher and the Coordinator of the Optical Transmission and Subsystems research line within the Optical Networks and Systems Department at the CTTC/CERCA, where she also serves as Project Management Coordinator. She received the M.Sc. degree in Electronics Engineering and the Ph.D. degree in Telecommunications Engineering from University Roma Tre, Rome, Italy, in 2003 and 2007, respectively. Her research interest areas include optical/digital signal processing and advanced transmission technologies for future optical networks.

Dr. Ramon Casellas holds the position of Senior Researcher within the Optical Networks and Systems Dept. at the CTTC/CERCA. After completing a double degree program from UPC (Spain) and ENST (France), he received a PhD de-

gree from ENST in 2002, where he later worked as an Associate Professor until 2005. His research interests include Optical Networks, Traffic Engineering and Control and Management architectures and protocols.

Dr. Ricardo Martínez holds the position of Senior Researcher within the Optical Networks and Systems Dept. at the CTTC/CERCA. He received a PhD degree in Electrical Engineering from UPC in 2007. His research interests include control and management of packet and optical networks as well as the convergence between mobile and fixed network infrastructures.

Dr. Monica Navarro is a Senior Researcher at CTTC/CERCA, where she holds the position of Head of the Communication Systems Division. She received the MSc degree in Telecommunications Engineering from UPC in 1997 and the PhD degree in Telecommunications from the Institute for Telecommunications Research (ITR), UniSA Australia, in 2002. She has also been part-time lecturer at the Universitat Pompeu Fabra (UPF), Barcelona. Her primary areas of interest are on information processing with applications to wireless communications and positioning.

Mr. Francisco Javier Vílchez holds the position of Researcher within the ONS Department and serves as Coordinator of the ONS laboratory at CTTC/CERCA. He received the MSc degree in Telecomunication Engineering and in Electronic Engineering in 2002 and 2007, respectively, from UPC.

Dr. Raül Muñoz is a senior researcher and head of the Optical Networks and System Department at CTTC/CERCA. He graduated and received a Ph.D. in telecommunications engineering in 2001 and 2005, respectively, from UPC (Spain). His research interests include control and management architectures, protocols and traffic engineering algorithms for future transport networks.

Dr. Ricard Vilalta is Senior Researcher at CTTC/CERCA. received the telecommunications engineering degree (2007) and Ph.D. degree (2013), at UPC, Spain. He is currently a Research Associate at Open Networking Foundation. His research is focused on SDN/NFV, Network Virtualization and Network Orchestration.

Dr. Laia Nadal holds the position of Researcher within the Optical Networks and Systems Department at CTTC/CERCA. She received the PhD degree in Signal Theory and Communications from UPC in 2014. Her research interests include signal processing and advance modulation formats for optical communication systems.

Mr. Arturo Mayoral graduated in Telecommunications Engineering by the Universidad Autónoma de Madrid in 2013, now is a Ph.D. student within the Department of Signal Theory and Communications (TSC) at UPC. In September 2013, he joined CTTC/CERCA as Software Engineer and Research Assistant in the Communications Network Division.

ABSTRACT

This paper presents a novel system architecture and describes a proof-of-concept for a virtualized GNSS receiver. The main idea consists of moving apart the GNSS antenna from the baseband processing, thus allowing for a number of new applications and services based on GNSS signal products. After contextualizing this concept into a major trend followed by the IT industry and the academia in the latter years, the paper provides a snapshot on the latest industry trends and describes a vibrant ecosystem of open source projects and tools, which make possible the actual (and rapid) deployment of virtual services over software-defined optical networks that allow for the continuous transmission of GNSS signals from the antenna to a remote baseband processor. After providing a brief description of the main concepts and building blocks of the presented architecture, and identifying the fundamental technological bottlenecks, we present a proof-of-concept implemented with actual commercial–off–the–shelf hardware components, providing evidences of the system feasibility and releasing different versions of a free and open source software-defined, *virtual* GNSS receiver.

1. INTRODUCTION

In recent years, the Information Technologies (IT) industry has witnessed an explosion of successful new business models, services and applications based on the combination of three key concepts: the cloud, software defined networking and network function virtualization. Namely, the *cloud* is a term referring to accessing computer, IT, and software applications through a network connection, often by accessing data centers using wide area networking (WAN) or Internet connectivity; *Software-defined networking* (SDN) is an approach to computer networking that allows network administrators to programmatically initialize, control, change, and manage network behavior dynamically via open interfaces and abstraction of lower-level functionality; and *Network functions virtualization* (NFV) is a network architecture concept that uses technologies for logically dividing the system's IT resources to implement *virtual* network node functions (*i.e.*, functional entities that act as physical devices) which can then be managed as building blocks that may be connected, or chained together, to create communication services.

The application of such concepts and associated technologies allows companies to increase the efficiency and flexibility of their IT resources by managing them as logical entities instead of as physical, hardwired units dedicated to a given application

or service. Examples of industry-embraced, commercial success stories are: *i*) the Cloud Radio Access Network (C-RAN) approach of mobile communication network operators [1], in which the baseband unit is moved away from physical antennas, allowing the virtualization of mobile base stations; *ii*) commercial Platforms as a Service (PaaS) [2], in which consumers deploy their own applications using the computing platform supported by the provider, such those offered by Amazon Web Services Elastic Beanstalk, Windows Azure, Heroku, IBM Bluemix or Google App Engine; *iii*) commercial Infrastructures as a Service (IaaS) [3], which offers processing, storage and fundamental computing resources to consumers so they can run their own applications, but without control of the underlying infrastructure (such as those offered by Amazon Web Services EC2 and S3, Windows Azure, Rackspace, Google Compute Engine and others); and *iv*) content delivery networks (CDN) [4] that serve content to end-users with high availability and high performance, such as those offered by Quantil, Limelight and Cloudflare, among others. In all those successful use cases, SDN and NVF concepts play a crucial role in their inception, design, implementation and commercial deployment.

The trend to programmatically define functions that were traditionally implemented in dedicated hardware devices has also reached Global Navigation Satellite Systems (GNSS), with a number of software-defined receivers currently available both at open source and commercial levels. A software-defined GNSS receiver is a computer program that takes raw GNSS signal samples as its input and performs all the baseband processing up to the computation of GNSS observables and the Position-Velocity-Time (PVT) solution, thus replacing dedicated integrated circuits. In general, these programs can work in post-processing mode (for instance, from raw signal samples stored in a file) or, with enough computational power, in real-time mode. The latter requires a GNSS antenna and a radio-frequency front-end performing signal amplification, frequency downshifting, filtering and conversion into the digital domain. Then, the delivered stream of raw signal samples is connected to a computer (via USB, Gigabit Ethernet, PCIe Express or optical fiber cable), where the software-defined receiver executes the baseband processing chain. First instances of software-defined GNSS receivers appeared in academia (see for instance TRIGR, from the Ohio University [5]; NAMURU, from the University of New South Wales [6]; N-GENE, from Istituto Superiore Mario Boella (ISMB) and Politecnico di Torino [7]; and GSNRx, from the University of Calgary [8]), and currently there are commercial options (such as SX3 [9], Aramis [10], Piksi [11], GSN and Nusar), and projects released under free and open source licenses (*e.g.*, OpenSource GPS [12], GPS-SDR [13], SoftGNSS [14], GNSS-SDRLIB [15] and GNSS-SDR [16]).

A software-defined GNSS receiver executed in the cloud appears to be the next natural step in this technology trend. However, the particularities of GNSS signal processing pose some technical challenges (described below) to an actual system deployment. In fact, other cloud-based solutions for software-defined GNSS receivers have already been proposed in academic literature. All of them are based on a "snapshot-based GNSS receiver" (methods proposed for instance in [17, 18, 19]), in which a batch of data (that could be as short as 2 ms of signal) is sent to the software receiver, which is able to compute a PVT solution with the aid of pre-loaded ephemeris. Notable examples of such approach are Microsoft's energy-efficient GPS sensing with cloud offloading [20, 21] and the system running on Amazon Web Services presented in [22, 23]. These approaches allow extremely low power consumption to the user equipment, at the expense of limited accuracy (ranging from 30 to 100 m of error for the 80 percent of locations) and high latency.

In this paper, we propose a different approach consisting of a software-defined, *virtualized* GNSS receiver, executed in the cloud, receiving a collection of GNSS signals' streams captured by a set of radio heads located elsewhere, and connected to the cloud via a high-performance communication network. The proposed system architecture allows for *continuous* GNSS signal streaming from the antenna to the GNSS baseband unit, in addition to any arbitrary duty cycling required by a certain application or service. The possibilities of such a system architecture are enormous. For instance, one could envisage the rapid deployment of a set of low-cost radio heads sending raw GNSS signals to a software-defined receiver in the cloud, and thus acting as a network of GNSS reference stations, generating high-rate pseudorange, phase and Doppler observables at real-time, and interesting products such as differential data to provide high-accuracy services to third users. This architecture allows for a number of interesting applications and services, outlined in Section 2.6, in addition to those already proposed for snapshot-based GNSS cloud offloading.

The main technical challenge is system availability: the transport network and the cloud infrastructure must ensure that the system is in functioning condition 100% of the time. This means high bandwidth, low latency, and possibly QoS and encryption mechanisms in the transport network, and high computational capabilities for the cloud infrastructure: the host device executing the software-defined receiver must reach real-time, which is not obvious in the most demanding configurations (*i.e.*, multi-system, multi-band receivers). Another key aspect for a real deployment is scalability, which is how such a system adapts to more users, more GNSS signals and bands, more external data sources or to more complex signal and data processing algorithms. In some applications, other aspects such as reliability (a trusted receiver for certification / security aspects), efficiency (power consumption trade-off between the user equipment and the cloud infrastructure), interoperability (the possibility

to exchange information with other sources, devices and systems) and marketability (for instance, providing a rapid path from a source code change in the software receiver to service deployment) could be of equal importance.

Leveraging on the practices and trends of the IT industry described above, this paper proposes a system architecture which addresses such technical challenges by defining a virtual network function infrastructure for a new generation of virtualized software-defined GNSS receivers and the new applications and services that such a concept makes possible. The remainder of the paper is organized as follows. Section 2 describes the proposed system architecture, Section 3 describes the proof of concept, and Section 4 concludes the paper. For the sake of the reader's convenience, two appendices have been included at the end of the paper, including a list of acronyms (Appendix A) and a taxonomy of software projects (Appendix B) mentioned along the text.

2. SYSTEM DEFINITION

The proposed system is depicted in Figure 1. This section defines the functionality of the building blocks and provides an overview of existing industry-grade implementations.

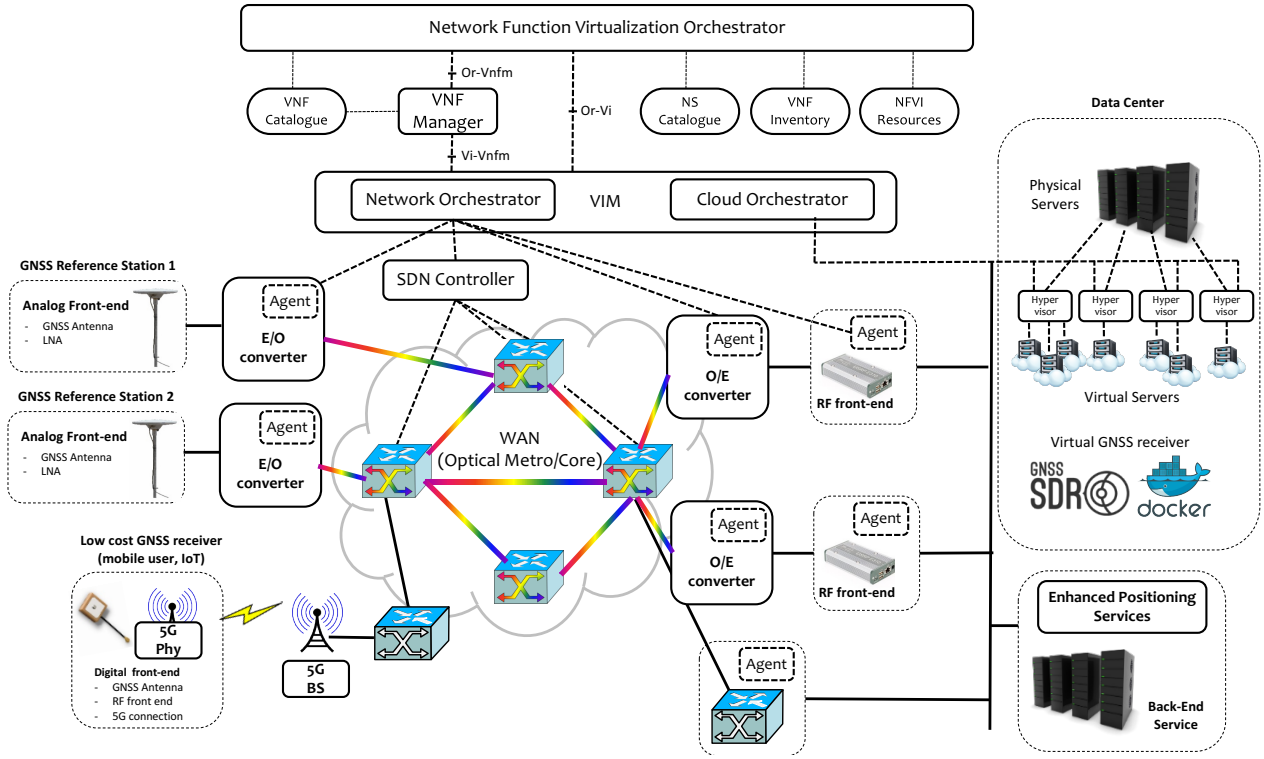

Fig. 1. Block diagram of the whole system.

2.1. End-to-end Network Function Virtualization management and orchestration (NFV - MANO)

The automated arrangement, coordination, and management of computer systems, middleware, and services, generally referred to as *orchestration*, has emerged as a key concept in such a software-defined-everything environment. Orchestration is understood as the coherent coordination of heterogeneous systems, allocating diverse resources and composing functions to offer end-user services. The standardization efforts are lead by NFV MANO, a working group of the European Telecommunications Standards Institute Industry Specification Group (ETSI ISG NFV). It is the ETSI-defined framework for the management and orchestration of all resources in the cloud data center. This includes computing, networking, storage, and virtual machine resources. Founded in November 2012 by seven of the world's leading telecoms network operators, this large community (currently composed of more than 290 companies including 38 of the world's major service providers), is still working intensely

to develop the required standards for NFV. The most recent sets of normative documents, known as releases, are described in [24, 25].

In parallel, the NFV market is evolving rapidly, as technology and service providers alike use pieces of open source and add them to their existing software solutions to provide the full complement of NFV functions, resulting in a vibrant ecosystem of free and open source software projects that is constantly progressing and growing, thus proving a large number of options with different levels of maturity, functional coverage and adherence to the ETSI-proposed model. In September 2014, the Linux Foundation announced the Open Platform for NFV Project (OPNFV), which facilitates the development and evolution of NFV components across various open source ecosystems. OPNFV integrates components from upstream projects (such as OpenStack, OpenDaylight, ONOS, OpenContrail, Ceph, KVM and LXD in the control plane; and FD.io, Open vSwitch, DPDK and OpenDataPlane in the data plane) across compute, storage and network virtualization in order to provide an comprehensively tested, industrially-proven end-to-end reference NVF platform. Specially relevant is OpenStack, a cloud management system consisting of interrelated components that control diverse, multi-vendor hardware pools of processing, storage, and networking resources throughout a data center, managed through a dashboard or via the OpenStack application program interface. The OpenStack community sustains a broad set of projects, each one providing specific services (*e.g.*, Nova manages computing instances, Neutron enables network connectivity, Swift manages data storage, Keystone provides authentication and authorization services, etc.), from which the OPNFV community consumes a sub-set. In should be noted that, in general, open source projects follow roadmaps driven by service requirements, and they not always can be clearly mapped into a reference architecture functionality [26].

In addition to OPNFV, a plurality of solutions is already available. Notably, Open Source MANO (OSM) [27], an ETSI-hosted project to develop an open source NFV Management and Orchestration software stack aligned with ETSI MANO. Other examples of open source ETSI NFV compliant MANO frameworks are Open Baton, Open-O and Cloudify. In February 2017, the Linux Foundation announced the creation of the Open Network Automation Platform (ONAP) project, aimed to create a harmonized and comprehensive framework for software automation of virtual network functions that expands the scope of ETSI MANO. The full picture results in an incommensurable number of possible combinations and configuration options, providing a wide choice of tools for the automated deployment of network services of very different nature and requirements. This trend towards the full automation of provisioning processes, along with a generalized orchestration of systems, is relying on diverse initiatives combining *de facto* and *de jure* open, modular and extensible standards, including the fast prototyping, open development, user-driven and frequent releases of open source projects and initiatives [26]. For more details about the software tools mentioned along this paper, the reader is referred to Appendix B.

The ETSI NFV MANO architectural framework defines three main functional blocks [28]: the NFV orchestrator, the virtual network function manager (VNFM) and the virtualized infrastructure manager (VIM), which are depicted in the upper part of Figure 1, as well as *reference points* (denoted in Figure 1 as "Or-Vnfm", "Vi-Vnfm" and "Or-Vi"), which are peer-to-peer information exchange mechanisms within functional blocks, following a producer-consumer paradigm and through well-defined APIs. The role of those blocks is described hereafter.

2.1.1. Network Function Virtualization Orchestrator

The Network Function Virtualization Orchestrator (NFVO) is a functional block within the MANO framework that is responsible for on-boarding of new network services and virtual network function (VNF) packages, network services lifecycle management, global resource management, and the validation and authorization of network functions virtualization infrastructure (NFVI) resource requests. The NFVI is the totality of the hardware and software components which build up the environment in which VNFs are deployed. This includes the user equipment, the network elements all the compute, storage, and networking resources. A Virtual Network Function (VNF) is defined as a functional block that has well-defined external interfaces and well-defined functional behavior, and that can be deployed in a NFVI.

Resource orchestration is important to ensure there are adequate compute, storage, and network resources available to provide a network service. To meet that objective, the NFVO can work either with the VIM or directly with NFVI resources, depending on the requirements. It has the ability to coordinate, authorize, release, and engage NFVI resources independently of any specific VIM. It also provides governance of VNF instances sharing resources of the NFVI.

The NVFO also maintains four repositories: two catalogues that hold the information related to the creation and management of the all the supported network services and VNF packages, a third repository holding information of all VNF and Network

Service instances, and a NFVI Resources repository holding information about available/reserved/allocated NFVI resources as abstracted by the VIM.

Today, in 2017, NFV orchestrators are typically delivered as a component of an NFV MANO solution, as is the case in the aforementioned Open Source MANO, OPNFV, Open-O or Open Baton frameworks.

2.1.2. Virtual Network Function Manager

The Virtual Network Function Manager (VNFM) is a functional block within the MANO framework that is responsible for the lifecycle management of VNF instances. This includes operations such as VNF instantiation (that is, to create a VNF using the VNF on-boarding artifacts); VNF scaling in/out (that is, the ability to scale by adding/removing resource instances, for instance virtual machines); scaling up/down (the ability to scale by changing allocated resources, *e.g.* increasing/decreasing memory, CPU capacity or storage size); updating and/or upgrading (support VNF software and/or configuration changes of various complexity); and VFN termination (release of VNF-associated NFVI resources). Other VNFM functions include VNF initial configuration (*e.g.*, assigning IP addresses), instantiation feasibility checking, notification of changes in the VNF lifecycle, integrity monitoring, and the collection of VNF instance-related NFVI performance measurement results.

Large players such as Nokia, Cisco, Ericsson, Huawei and NEC/Netcracker all have VNFM offerings, which are sometimes offered as a component of an overall NFV MANO solution. In the open source community, Tacker is an OpenStack project that consists of an open NFV orchestrator with a built-in, general-purpose VNF manager to deploy and operate virtual network functions on an NFV platform.

2.1.3. Virtualized Infrastructure Manager (VIM)

The Virtualized Infrastructure Manager (VIM) is the functional block within the MANO framework that is responsible for controlling, managing and monitoring the NFVI compute, storage, and network resources. There can be a single VIM or multiple, specialized VIMs (*e.g.*, compute-only, storage-only, networking-only), but the idea is to have a single abstraction layer that exposes northbound open interfaces that support management of the NFVI, and southbound interfaces that interact to a variety of network controllers and hypervisors (that is, programs that create and run virtual machines) in order to perform the functionality exposed through its northbound interfaces. Hence, this provides VNF managers and NFV orchestrators with the ability to deploy and manage VNFs.

This block keeps an inventory of the allocation of virtual resources to physical resources. This allows the VIM to orchestrate the allocation, upgrade, release, and reclamation of NFVI hardware resources (compute, storage, networking) and software resources (*e.g.*, hypervisors). It also collects performance and fault information, which allows to exercise usage optimization.

In the diagram shown in Figure 1, we distinguish two functional sub-blocks within the VIM: a network orchestrator, in charge of managing the end-to-end connectivity (i.e., the communication network from the user equipment to the compute resources executing instances of a software-defined GNSS receiver, being a private data center, a public cloud computing service, or a mix of both), and a cloud orchestrator, specialized in hardware and software resources.

- The **network orchestrator** supports the end-to-end management of VNF Forwarding Graphs, *e.g.* by creating and maintaining virtual links, virtual networks, sub-nets, and ports, in order to transport the GNSS signals collected by the user equipment through the communication network to a data center, in which a computer will be executing one or more virtual machines or software containers, one of them executing the software-defined GNSS receiver that will process the signals gathered by the user. The network orchestrator sends requests to the *agents* of the user equipment and back-end network end-points. An agent is a program continuously running as a background process (sometimes called a "daemon") at each of those elements that listens for such requests and applies the corresponding actions. The network orchestrator is also in charge of the management of security group policies to ensure network/traffic access control.

- The **cloud orchestrator** coordinates the server hardware, so that virtual server instances (*e.g.*, virtual machines or software containers), can be created from the most convenient underlying physical server. It can be used to manage a range of virtual IT resources across multiple physical servers, and provides for centralized administration of virtualized resources including creating, storing, backing up, patching and monitoring. It is also in charge of the management of software images (*e.g.*, a virtualized GNSS receiver) as requested by the NFVO and the VNFM.

The OPNFV community consumes a sub-set of OpenStack projects to implement VIM. In turn, OpenStack implements a sub-set of the functionalities that ETSI could end up defining for VIMs. Other open source examples are OpenNebula and OpenVIM, now integrated into Open Source MANO. As commercially available VIM tools, we can mention VMware vCloud Director and Citrix XenServer. Many other vendors supply VIM solutions as add-ons to OpenStack. If software containers are used instead of virtual machines (more details in Section 2.5.3), there are container orchestrators such as Docker Swarm, Kubernetes, Nomad and Mesosphere Enterprise DC/OS. Compute resources orchestration can include remote cloud hosting services, such as those offered by Amazon Elastic Compute Cloud (EC2), Google Compute Engine and Microsoft Azure.

2.2. Software Defined Networking Controller

A SDN Controller is the application that acts as strategic control point in the SDN network, managing the flow control to the switches/routers "below" in Figure 1 (via the so-called southbound APIs) and the applications and business logic "above" (via northbound APIs) to deploy intelligent networks. In the northbound direction, the control plane provides a common abstracted view of the network to higher-level applications and programs using APIs. In the southbound direction, the control plane programs the forwarding behavior of the data plane, using device level APIs of the physical network equipment distributed around the network. The SDN Controller is then in charge of managing the network elements (switches, routers, etc.) that will transport the GNSS signal streams from users equipment to the compute resources executing instances of virtualized GNSS receivers.

The OpenFlow protocol [29], considered the first SDN standard, defines the open communications protocol that allows the SDN Controller to work with the forwarding plane (also known as data or user plane, defined as the part of the router architecture that decides what to do with packets arriving on an inbound interface) and make changes to the network. In particular, one of the OpenFlow strengths is to have identified a basic common hardware model for a data switch (with emphasis in packet switching) and, by virtue of the formal specification of tables, flows, and actions, to provide a flexible and extensible mechanism to automate networking decisions. However, it also poses some challenges in terms of scalability, security and the need of specialized hardware, so other alternatives to OpenFlow are emerging such as the Border Gateway Protocol (BGP), NETCONF, Extensible Messaging and Presence Protocol (XMPP), Open vSwitch Database Management Protocol (OVSDB) and the Multiprotocol Label Switching Transport Profile (MPLS-TP).

There are many SDN controllers available today relevant to virtual environments (*e.g.*, Neutron), an OpenStack project that enables network connectivity as a service between interface devices (*e.g.*, virtualized Network Interface Cards, VNICs) managed by other OpenStack services (*e.g.*, Nova); OpenDaylight (the industry's *de facto* SDN platform, with more than 100 deployments, including Orange, China Mobile, AT&T, T-Mobile, Comcast, KT Corporation, Telefonica, TeliaSonera, China Telecom, Deutsche Telekom, and Globe Telecom), ONOS (a carrier-grade SDN network operating system) and OpenContrail (an extensible platform for SDN).

2.3. User equipment

2.3.1. Radio heads

We distinguish two types of users:

- Type A: Static users with connectivity to an optical network. In this scenario, the user equipment (that is, the radio head) would consists of one or more GNSS antennas, each one with one Low Noise Amplifier per targeted GNSS band (see Table 1), an Electrical-to-Optical (E-O) converter, and an optical fiber connection to an optical network. This approach is usually known as radio-over-fiber (RoF) [30, 31, 32].

- Type B: Mobile users (or users without connection to an optical network). In this scenario, the user equipment would consist of one or more GNSS antennas, each one with a RF front-end per targeted GNSS band that converts the RF (analogue) signal to a stream of digitized signal samples, usually downconverted to baseband or to a low intermediate frequency. The analog-to-digital conversion will deliver a minimum data bit rate of

$$r \geq BW \cdot 2 \cdot q \cdot N, \tag{1}$$

  where BW is the targeted (passband) bandwidth, q is the number of bits per sample and N is the number of antennas. The data stream(s) must be then sent to the network through a wireless interface. Table 1 provides some figures of the required data rate for different GNSS receiver configurations. Moreover, a 20% of overhead caused by PHY/MAC/Net

protocols should be added to those rates. By simple inspection, it is easy to see that a LTE network (which theoretically sustains a peak upload rate of 50 Mbps, but on average allows about 1.5 Mbps [33]) cannot cover the more basic receiver configuration for a continuous signal delivery. This is actually the bottleneck that snapshot-based cloud GNSS receivers overcome by operating on discontinuous segments of GNSS signal. Even LTE-A/4G networks (which theoretically can sustain peak upload rates of 500 Mbps, but in practice averaging about 12 Mbps [33]) do not meet the requirements of most challenging configurations. Current designs of 5G networks are targeting upload rates possibly up to 10 Gbps and guaranteed data rates up to 50 Mbps [34], which would cover most of the possible GNSS frequency plans. Other possible wireless technologies for this link could be IEEE 802.11n or 802.11ac, which typically deliver an average throughput of 35 Mbps and 250 Mbps per data stream, respectively.

2.3.2. Agents

Network agents monitor network resources and make IP addresses and computer names available. They can be from simple scripts to complex, full-featured software tools, depending on the service or application requirements.

2.4. Network elements

2.4.1. Network routers and switches

A router is a networking device that forwards data packets between computer networks. They operate at the network layer, and its role is to connect different logical sub-networks (*e.g.*, 5G to WAN, which in Figure 1 inject GNSS signals gathered by mobile users to the optical network; or WAN to LAN, which in Figure 1 inject GNSS signals from the optical network to the local area network of the data center). Devices operating at a data link layer (for instance connecting devices within a LAN, or nodes within an optical WLAN) are usually referred to as switches.

In a classical switch based on the Ethernet protocol, the fast packet forwarding (data path) and the high level routing decisions (control path) are managed on the same device. On the contrary, an OpenFlow-based switch separates these two functions. The data path portion still resides on the switch, while high-level routing decisions are moved to a separate controller (*e.g.*, the SDN controller in Figure 1). The OpenFlow switch and controller communicate via the OpenFlow protocol, which defines messages such as “packet-received”, “send-packet-out”, “modify-forwarding-table”, and “get-stats”. Other alternative protocols to OpenFlow operate under the same principle.

2.4.2. Optical switches

The data traveling in the form of light (possibly on multiple wavelengths) through an optical network need to be switched at the network nodes. A data stream arriving at a given node is forwarded to its final destination via the best possible path, which is determined by factors such as distance, cost, and the reliability of specific routes. While conventional optical switching consists of converting the input fiber optical signal to an electrical signal, performing the switching in the electrical domain, and then converting the electrical signal back to an optical signal that goes down the desired output fiber, new approaches such as reconfigurable optical add/drop multiplexer (ROADM) systems are able to avoid the unnecessary O-E-O conversion (and its associated expensive, bulky, and bit-rate/protocol dependent subsystems), enabling *transparent* O-O-O systems that use optical switching. This involves lower capital expenditures (CAPEX), as there is no need for a large amount of expensive high-speed electronics, and lower operational expenditures (OPEX) because fewer network elements are required. The complexity reduction also allows for physically smaller optical switches.

There are four main types of ROADM: Type I, with fixed (colored) ports; Type II, which offers reconfigurable (colorless) add/drop ports; Wavelength Selective Switches (WSS), that allow for degree-N connectivity; and Optical Cross-Connects (OXC), which are used for wavelength cross-connect switching in mesh networks. The Generalized Multi-Protocol Label Switching (GMPLS) [36] is the *de facto* control plane of wavelength switched optical networks.

2.5. Back-end

2.5.1. Signal ingestion

In the case of Type A users (those that inject the RF signal received at the GNSS antenna directly into a fiber via an E-O converter), the signal must be converted back to the electric domain via an O-E converter. Then, the analogue signal stream

Band	Center Freq. (MHz)	Service	GNSS Signal	Mux	Modulation	Min. required bandwidth BW_{min}	Receiver reference bandwidth BW_{ref}	Minimum received power at BW_{ref}	Min. bit rate with $q = 2$ at BW_{min}	Min. bit rate with $q = 8$ at BW_{ref}
	1176.45	OS/SoL	GPS L5$^{(*)}$	CDMA	BPSK(10)	20.46 MHz	25 MHz	-157.9 dBW	81.84 Mbps	400 Mbps
	1176.45	OS	Galileo E5a	CDMA	∼QPSK(10)	20.46 MHz	20.460 MHz	-155 dBW	81.84 Mbps	327.36 Mbps
	1176.45	OS	GLONASS L5OCM$^{(**)}$	CDMA	BPSK(10)	N/A; ∼20 MHz	N/A	N/A	∼80 Mbps	N/A; >320 Mbps
	1191.795	PRS	Galileo E5	CDMA	AltBOC(15,2.5)	N/A; ∼50 MHz	51.150 MHz	N/A	∼200 Mbps	818.4 Mbps
E5	1202.025	OS	GLONASS L3OC$^{(*)}$	CDMA	BPSK(10)	20.46 MHz	N/A	-158.5 dBW	81.84 Mbps	N/A; >327.36 Mbps
	1207.14	OS/SoL	Galileo E5b	CDMA	∼QPSK(10)	20.46 MHz	20.460 MHz	-155 dBW	81.84 Mbps	327.36 Mbps
	1207.14	OS	Beidou B2I	CDMA	BPSK(2)	N/A; ∼4 MHz	36 MHz	-163 dBW	16 Mbps	576 Mbps
	1207.14	M	Beidou B2Q	CDMA	BPSK(10)	N/A; ∼20 MHz	36 MHz	N/A	∼80 Mbps	576 Mbps
	1207.14	OS	GLONASS L3OCM$^{(**)}$	CDMA	BPSK(10)	N/A; ∼20 MHz	N/A	N/A	∼80 Mbps	N/A; >320 Mbps
	1227.60	OS	GPS L2C$^{(*)}$	CDMA	BPSK(1)	2.046 MHz	30.69 MHz	-161.5 dBW	8.184 Mbps	491.04 Mbps
	1227.60	M	GPS P	CDMA	BPSK(10)	20.46 MHz	20.46 MHz	-161.4 dBW	81.84 Mbps	327.36 Mbps
	1227.60	M	GPS M$^{(*)}$	CDMA	BOC(10,5)	N/A; ∼24 MHz	30.69 MHz	N/A	∼96 Mbps	491.04 Mbps
L2	1246.00	OS	GLONASS L2OF	FDMA	BPSK(0.5)	6.7095 MHz	6.7095 MHz	-167 dBW	26.838 Mbps	107.352 Mbps
	1246.00	M	GLONASS L2SF	FDMA	BPSK(5)	15.907 MHz	N/A	N/A	63.63 Mbps	N/A; >254.52 Mbps
	1248.06	OS	GLONASS L2OC$^{(**)}$	CDMA	BOC(1,1)	N/A; ∼4 MHz	N/A	-158.5 dBW	∼16 Mbps	N/A; >64 Mbps
	1248.06	M	GLONASS L2SC$^{(**)}$	CDMA	BOC(5,2.5)	N/A; ∼15 MHz	N/A	-158.5 dBW	∼60 Mbps	N/A; >240 Mbps
	1268.52	M	Beidou B3	CDMA	BPSK(10)	20.46 MHz	20.460 MHz	N/A	81.84 Mbps	327.36 Mbps
E6	1278.75	CS	Galileo E6b/c	CDMA	BPSK(5)	10.23 MHz	40.920 MHz	-155 dBW	40.920 Mbps	654.72 Mbps
	1278.75	PRS	Galileo E6a	CDMA	BOC_{cos}(10,5)	N/A; ∼30 MHz	40.920 MHz	N/A	∼120 Mbps	654.72 Mbps
	1561.098	OS	Beidou B1	CDMA	QPSK(2)	4.092 MHz	16 MHz	-163 dBW	16.368 Mbps	256 Mbps
	1575.42	OS	GPS C/A	CDMA	BPSK(1)	2.046 MHz	20.46 MHz	-158.5 dBW	8.184 Mbps	327.36 Mbps
	1575.42	M	GPS P	CDMA	BPSK(10)	20.46 MHz	20.46 MHz	-161.5 dBW	81.84 Mbps	327.36 Mbps
	1575.42	OS	GPS L1C$^{(**)}$	CDMA	BOC(1,1)	4.092 MHz	30.69 MHz	-157 dBW	16.368 Mbps	491.04 Mbps
	1575.42	M	GPS M$^{(*)}$	CDMA	BOC(10,5)	N/A; ∼24 MHz	30.69 MHz	N/A	∼96 Mbps	491.04 Mbps
	1575.42	OS	GLONASS L1OCM$^{(**)}$	CDMA	BOC(1,1)	N/A; ∼ 4 MHz	N/A	N/A	∼16 Mbps	N/A; >64 Mbps
L1	1575.42	PRS	Galileo E1a	CDMA	BOC_{cos}(15,2.5)	N/A; ∼40 MHz	N/A	N/A	∼160 Mbps	N/A; >640 Mbps
	1575.42	OS/SoL	Galileo E1b/c	CDMA	CBOC(6,1,1/11)	4.092 MHz	24.552 MHz	-157 dBW	16.368 Mbps	392.832 Mbps
	1600.995	OS	GLONASS L1OC$^{(**)}$	CDMA	BOC(1,1)	N/A; ∼ 4 MHz	N/A	-161.5 dBW	∼16 Mbps	N/A; >64 Mbps
	1600.995	M	GLONASS L1SC$^{(**)}$	CDMA	BOC(5,2.5)	N/A; ∼15 MHz	N/A	NA dBW	∼60 Mbps	N/A; >240 Mbps
	1602.00	OS	GLONASS L1OF	FDMA	BPSK(0.5)	8.3345 MHz	8.3345 MHz	-161 dBW	33.338 Mbps	133.352 Mbps
	1602.00	M	GLONASS L1SF	FDMA	BPSK(5)	17.5325 MHz	N/A	N/A	70.13 Mbps	N/A;>280.52 Mbps

Table 1. GNSS signals and their frequency allocation, as transmitted by satellites. The minimum required receiver bandwidth is computed upon its corresponding modulation and the Nyquist criterion (although narrower receivers are known to work, *e.g.*, for Galileo E5a and E5b [35]) and the reference bandwidth is as defined in the corresponding Interface Control Document. Notation is as follows: OS: Open Service; SoL: Safety of Life; CS: Commercial Service; PRS: Public Regulated Service; M: Military. CDMA/FDMA: Code / Frequency Division Multiple Access. BPSK(n): Binary Phase Shift Keying with chip rate $r_c = n \cdot 1.023$ Mcps. QPSK(n): Quadrature Phase Shift Keying with chip rate $r_c = n \cdot 1.023$ Mcps. BOC(n,m): Binary Offset Carrier with subcarrier rate $r_{sc} = n \cdot 1.023$ MHz and chip rate $r_c = m \cdot 1.023$ Mcps. N/A: Not Available. (*): Modernized signal not broadcast by all satellites. (**): Proposed signal.

(still at RF) must be converted down to baseband (or low IF), filtered and converted to the digital domain by an analogue-to-digital converter (ADC), and then sent to the corresponding host computer through the data center's LAN.

In the case of Type B users (those that inject digitized GNSS signals to the network), the data stream can be directly fed to the data center's LAN by a router connected to the optical WAN.

2.5.2. Data Center

A Data Center is a resource pool for storage, management, processing and distribution of data pertaining to a particular business or administrative domain. It is commonly understood as a (large) group of networked computer servers. Until recently, cloud management frameworks had been built assuming, in most cases, a centralized location, so servers could interchange information using Ethernet technologies within a LAN. However, now the concept has broadened to embrace distributed cloud infrastructures [26], where a set of either on-premises or remote, private or public computing clouds are all orchestrated together. This is specially interesting in services in which a low latency is required, for instance by placing computing resources near the end user.

2.5.3. Virtualized GNSS receivers

A virtualized software application is a program that can be executed regardless the underlying computer platform (*i.e.*, processor architecture, operating system and installed library versions) that is executing it. This can be achieved by packaging the application and all its software requirements (the operating system and all the application-required supporting libraries and programs) in a single, self-contained and isolated software entity, that can be then run on any platform. Hence, for instance, using virtualization tools a complete Windows system can be run on a Linux machine, or on another version of Windows. This is a very convenient strategy for orchestration, since it allows the elastic creation, execution and destruction of application instances as requested in a user basis, and to intelligently spread the running instances along the available compute resources, regardless of what they are exactly composed of (that is, processor architecture, version of the host operating system or physical location). An instance of a software-defined GNSS receiver executed in a virtual environment can then be called a *virtualized GNSS receiver*. There are two main approaches to software virtualization: virtual machines and software containers.

A virtual machine (VM) is a software-based environment designed to simulate a hardware-based environment, for the sake of the applications it will host. A VM emulates a computer architecture and provides the functionality of a physical computer. Within each virtual machine runs a full operating system, so conventional software applications expecting to be managed by an operating system and executed by a set of processor cores (*e.g.*, a software-defined GNSS receiver) can run within a VM without any required change. With VMs, a software component called a hypervisor interfaces between the VM environment and the underlying hardware, providing the necessary layer of abstraction. The hypervisor is responsible for executing the virtual machine assigned to it, and it can execute several of them simultaneously. Examples of hypervisors that a cloud orchestrator could control through an API server are the Kernel-based Virtual Machine (KVM), Xen, Quick Emulator (QEMU), VMware's ESXi, Oracle's VirtualBox and Microsoft's Hyper-V. Virtual machine technology enjoys a longstanding tradition in the IT industry, so there is plenty of commercial and open source choices, and load balancing mechanisms are well understood and established.

Recently, however, software containers are replacing VMs as the preferred supporting software stack system for virtualized software applications because of the faster and more lightweight nature of the former. An application running in a container can be more efficient in making use of the underlying hardware than when it is executed on a VM (since it operates directly with the real processing units instead of against an emulated layer, avoiding its overhead [37]), and many more containers than VMs can be put onto a single server, thus optimizing the investment in compute resources. The concept of containerization was originally developed as a mechanism to segregate namespaces in a Linux operating system for security purposes, isolating process groups (a process and possible descendant processes) from the outside world. The first approach consisted of producing partitions (sometimes called "jails") within which applications of questionable security or authenticity could be executed without risk to the kernel. The kernel was still responsible for execution, though a layer of abstraction was inserted between the kernel and the workload. Once the environment within these partitions was minimized for efficiency's sake, the concept expanded to make the contents of those partitions portable. Hence, this technology can be seen as an advanced implementation of the standard chroot mechanism in UNIX-like systems. The first container system was Linux Containers (LXC), followed by a container hypervisor (LXD), and then by other projects such as Docker or Ubuntu Snaps. These latter systems provide native environments with no hypervisor but a daemon that supplements the host kernel and that maintains the compartmentalization between containers,

while connecting the kernel to their workloads. Other solutions, such as Virtuozzo's, allow the creation of encrypted containers for security purposes.

2.6. Back-end services

The main services foreseen to have a natural accommodation in the system proposed in this paper are related to high accuracy positioning (allowing sophisticated, cm-error level algorithms such as PPP [38], Fast-PPP [39], network RTK [40] or WARTK [41]), rapid deployment of reference stations, GNSS signal authentication, and low energy consumption GNSS receivers. It follows a list of services and applications that the proposed system could make possible:

- A network of GNSS reference stations can produce differential data which can then be used to provide real-time corrections (and thus cm-level accuracy) to third users.
- Programmable output rate of GNSS observables, paving the way to new applications in space weather, precision agriculture, and surveying.
- Controlled reception pattern antennas (CRPAs) / Antenna arrays: Robustness to jamming / spoofing / multipath [42]. Localization of interference sources.
- Rapid deployment of GNSS-related infrastructure in disaster relief scenarios.
- Convenient solution for GNSS Commercial Services (*e.g.*, Galileo E6), GNSS Authentication, and security-related applications (GPS M code, Galileo PRS), since the encryption module remains on the service provider's premises.
- Certified "space-time-stamping". A user could grab a batch of GNSS signals, send it to the cloud, and receive back a trusted certificate of position and time.
- A low energy, cloud off-loaded GNSS receiver for the Internet of Things.

3. PROOF OF CONCEPT IMPLEMENTATION

In order to provide evidences which demonstrate the technical feasibility of the proposed concept, the authors implemented a proof-of-concept for users of Type A, as defined in Section 2.3 (that is, an static antenna with a direct connection to an optical network), in the simplest configuration of a single-antenna user. The experimental setup, shown in Fig. 8, was carried out by integrating two research facilities available at CTTC: GESTALT® (a GNSS signal testbed described at [43]) and the ADRENALINE Testbed® (an equipment for experimental research on high-performance and large-scale intelligent optical transport networks described at [44]). The arrangement consisted of a GNSS antenna platform at the roof of CTTC premises (see Fig. 2), whose signals were amplified and injected into an E-O converter in the user side; an optical transport network (including 35 km of optical fiber) emulating an optical WAN, in charge of transporting the GNSS signals, in form of light, from the user's antenna to the data center; and an O-E converter, a RF front-end and a virtualized GNSS receiver in the back-end side.

The user equipment consisted of an antenna, an amplification stage and an E-O converter. The chosen GNSS antenna was a NavXperience's 3G+C, which features a low noise amplifier providing a gain of 42 dB with noise figure of 2 dB. A 40-meter long 1/2" coaxial cable drove the RF signal from the antenna to a patch panel cabinet located in the lab and designed to protect the RF connectors from the weather effects, such as a lightning discharge. At the ending of the RF cable a second GNSS amplifier was placed (model A11 by GPS Source Inc.) providing 30 dB of gain with a noise figure of 1.8 dB, and its output connected to the E-O converter in charge of turning the received RF signals into light. The E-O converter consisted of a Tunics Reference SCL tunable laser source tuned at 1550.12 nm and with 2 dBm of output power; a Photline Technologies' DR-AN-40-MO single-ended driver (that is, a wideband RF non-inverting amplifier delivering a gain of 26 dB with a noise figure of 3 dB); and a Mach-Zehnder modulator which controlled the amplitude of the optical wave (see Fig. 3). The generated signal was then injected into an optical fiber that brought the signal to an optical switch that acted as the entry point of a transport optical network emulated with the ADRENALINE Testbed® (see Fig. 4).

The ADRENALINE Testbed® includes a multi-technology, software-defined control plane for multilayer (packet over optical) networks, which manages the networking resources and covers the long-haul core transport and aggregation segments, thus automating the processes involved in the provisioning of networking services, such as optical lightpaths or Ethernet/MPLS-TP/IP connectivity services. The design of ADRENALINE's control plane follows broad Software Defined Networking (SDN)

principles, such as stacking components in a hierarchical setting with different levels of abstraction. Network connectivity services are provisioned by an overarching control orchestration. In particular, at a given domain and layer, the control plane can based on the GMPLS technology and protocols – a distributed system in which a dedicated controller is responsible for each node autonomously – or follow SDN/OpenFlow principles, with a centralized controller that manages all the aspects of a network, dynamically configuring networks according to users an application needs. The facility also includes a SDN/NFV cloud computing platform, which can be deployed over multi-domain transport networks and distributed data centers [45].

In this particular experimental setup, the network management and orchestration was kept at the bare minimum. The optical network was configured as one OXC acting as the entry point, routing the optical signal through 35 km of real optical fiber (see Fig. 5), and with a ROADM Type II at the other end. The optical routing was performed by a self-developed GMPLS control plane, which is managed by a SDN controller implemented with OpenDaylight. The optical signal was then accessed from a port of the ROADM and converted again into the RF domain by means of a O-E converter (a Discovery Semiconductors DSC-R401HG InGaAs PIN photodetector with a transimpedance amplifier, *i.e.*, a current-to-voltage converter, delivering a linear response to $> +3$ dBm optical input, 600 mVp-p of linear output voltage, 20 GHz of RF bandwidth and a conversion gain of 160 V/W), directly connected to the antenna input of an Ettus Research's USRP E300 RF front-end (see Fig. 6). Such front-end downconverted the received GNSS RF signals into baseband and performed the Analog to Digital conversion. The stream of raw signal samples was then injected via Gigabit Ethernet to the computing resources, executing an instance of a virtual GNSS receiver and computing GNSS products and PVT fixes in real-time. The compute resources were managed with OpenStack (using Nova for controlling compute resources, Neutron for managing network connectivity as a service and Keystone for authentication). Fig. 7 shows the position of the antenna, as computed by the virtual GNSS receiver, as plotted by Google Earth. The receiver was also able to generate GNSS products (that is, pseudorange, phase-range and pseudorange rate observables) for GPS L1 C/A and Galileo E1b/c signals, obtained without any kind of external assistance nor differential system, and delivered in standard formats such as RINEX files or RTCM 10403.2 messages.

Three virtualization mechanisms were tried, all based on the free and open source software-defined receiver GNSS-SDR: *i*) GNSS-SDR running in a virtual machine prepared with VirtualBox; *ii*) GNSS-SDR in a Docker container; and *iii*) GNSS-SDR in an Ubuntu's Snap package. A full analysis of their performance is left as future work, but all of them reached real-time when using a sampling frequency of 4 Msps in the RF front-end. In the tests performed with live signals, we obtained a Circular Error Probability (that is, the radius of a circle centered at the average position, containing the position estimate with probability of 50%) of 2.16 m when using a simple least squares algorithm for the PVT computation. The creation of the GNSS virtual network function took 11.9 s, and the lightpath setup in the optical network took 4.2 s, as shown in Fig. 9. The latency measured from the antenna to the output of the O-E converter at the data center (see Fig. 10) was 177 μs, from which 116 μs correspond to the light propagation time along 35 km.

4. CONCLUSIONS

This paper introduced the concept of the virtualized GNSS receiver, in the context of network function virtualization for optical access networks. After an overview of the state-of-the-art and a description of the proposed system and related existing tools, including the identification of technology bottlenecks, we report the results of a proof-of-concept that demonstrates the feasibility of the physical separation of the GNSS antenna and the virtual GNSS receiver.

The proof-of-concept consisted of a transmission of GNSS RF signals (gathered by an antenna and directly converted into light) over optical fiber, transported through a software-defined optical network from the antenna to a remote (35 km away from the antenna) data center in which a virtualized GNSS receiver was performing the frequency downshifting, analog to digital conversion and all the baseband processing up to the generation of GNSS observables and PVT fixes, delivered in real time. The results obtained by this simple proof of concept shows the technical feasibility of the proposed approach using commercial-off-the-shelf optical and electronic components and the latest trends and available software tools from the IT industry. Other configurations (for instance, moving the RF front-end to the user side and transporting digitized data) are also possible in the presented setup, and will be explored in future contributions. A wireless user equipment is still a challenge due to the high throughput requirements of a continuous transmission of digitized GNSS signals.

An extensive list of existing tools for implementing management and network orchestration, network function virtualization, software-defined networking, virtual infrastructure management, and handling of virtualized GNSS receivers is provided in Appendix B, including the free and open source, virtualized GNSS receivers presented in this work.

Fig. 2. GNSS antennas at the roof of CTTC's headquarters.

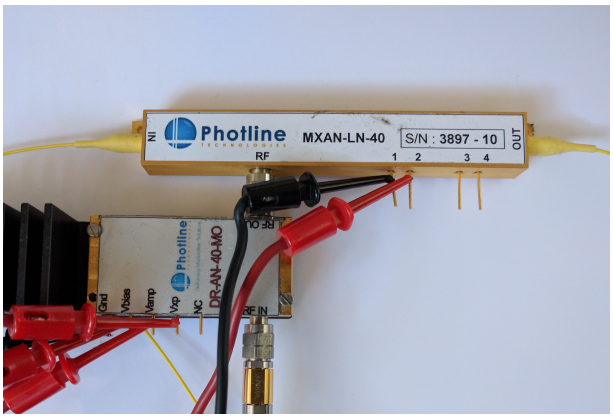

Fig. 3. E-O converter: driver and Mach-Zehnder modulator.

Fig. 4. Picture of ADRENALINE®'s Control / Data Plane.

Fig. 5. 35 km of optic fiber in coils.

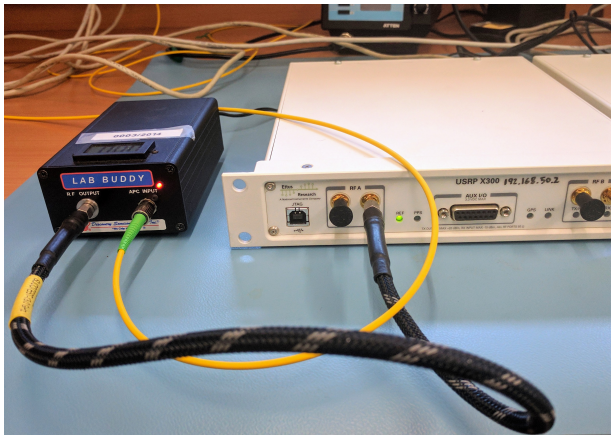

Fig. 6. O-E converter (left) and RF front-end (right).

Fig. 7. Locating the antenna with the virtual GNSS receiver.

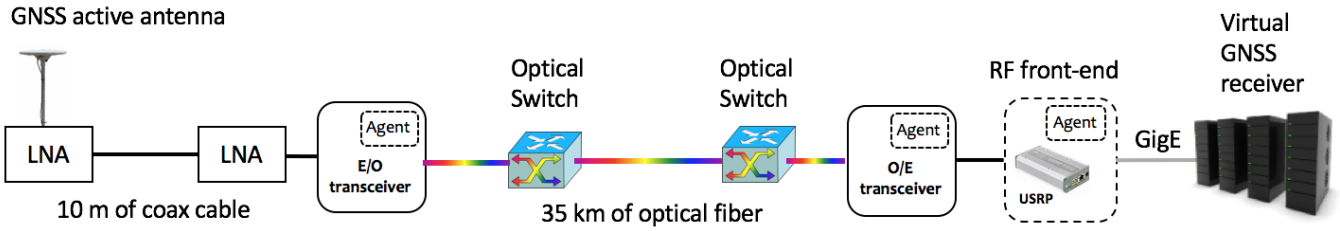

Fig. 8. Diagram of the Proof of Concept.

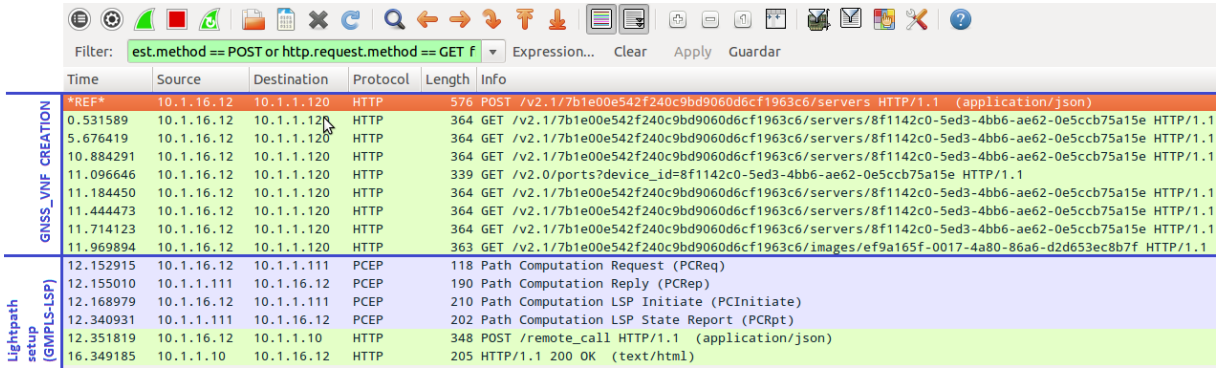

Fig. 9. Network packets traces as captured by Wireshark, a network protocol analyzer.

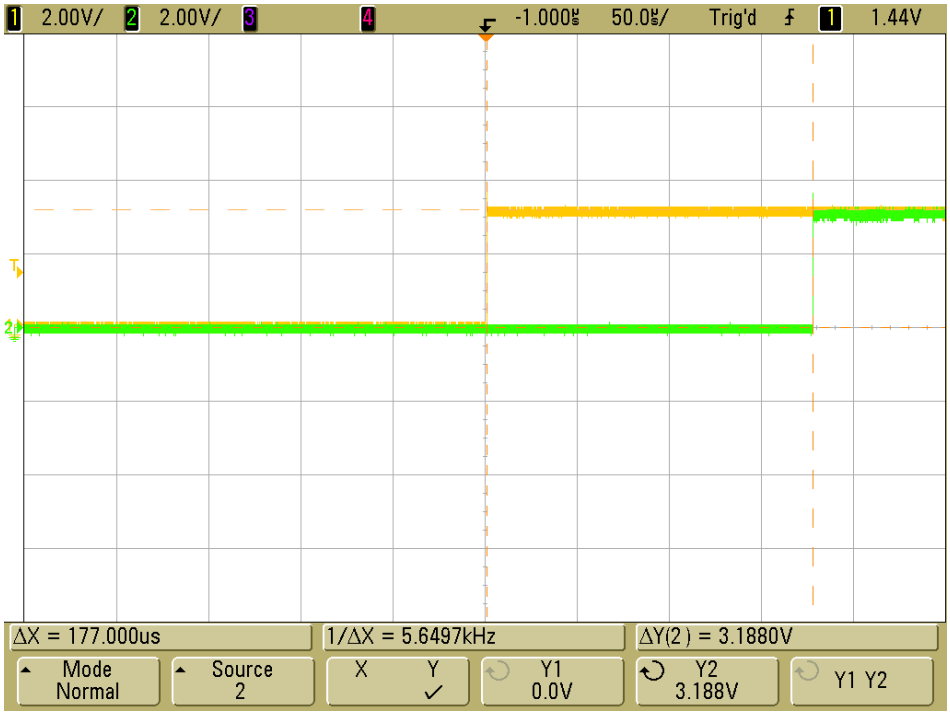

Fig. 10. The latency measured from the antenna to the output of the O-E converter was 177 μs.

A. APPENDIX: LIST OF ACRONYMS

Acronym	Definition
ADC	Analog to Digital Converter
AltBOC	Alternative Binary Offset Carrier
API	Application Program Interface
BOC	Binary Offset Carrier
BPSK	Binary Phase Shift Keying
CAPEX	Capital Expenditures
CBOC	Composite Binary Offset Carrier
CDMA	Code Division Multiple Access
CDN	Content Delivery Network
C-RAN	Cloud Radio Access Network
ETSI	European Telecommunications Standards Institute
E-O	Electrical to Optical
FDMA	Frequency Division Multiple Access
GMPLS	Generalized Multi-Protocol Label Switching
GNSS	Global Navigation Satellite Systems
IaaS	Infrastructure as a Service
IF	Intermediate Frequency
ISG	Industrial Specification Group
IT	Information Technologies
LAN	Local Area Network
MANO	Management and Orchestration
NFV	Network Function Virtualization
NVFI	NVF Infrastructure
NVFO	NVF Orchestrator
OCX	Optical Cross-Connect
O-E	Optical to Electrical
O-E-O	Optical to Electrical to Optical
⋮	⋮
⋮	⋮
O-O-O	Optical to Optical to Optical
OPEX	Operational Expenditures
OPNFV	Open Platform for NFV Project
PaaS	Platform as a Service
PCIe	Peripheral Component Interconnect Express
PIN	Positive-Intrinsic-Negative
PPP	Precise Point Positioning
PVT	Position-Velocity-Time
QoS	Quality of Service
QPSK	Quadrature Phase Shift Keying
RF	Radio Frequency
RINEX	Receiver Independent Exchange Format
ROADM	Reconfigurable optical add/drop multiplexer
RTCM	Radio Technical Commission for Maritime Services
RTK	Real Time Kinematics
SDN	Software Defined Networking
SDR	Software Defined Radio
USB	Universal Serial Bus
VIM	Virtualized Infrastructure Manager
VM	Virtual Machine
VNF	Virtual Network Function
VNIC	Virtualized Network Interface Card
WAN	Wide Area Network
WARTK	Wide Area Real Time Kinematics
WSS	Wavelength Selective Switches

B. APPENDIX: SOFTWARE PROJECTS

B.1. NVF-MANO frameworks

Name	Description
OPNFV	Reference NFV platform. `https://www.opnfv.org`
Open Source MANO (OSM)	ETSI-hosted project to develop an Open Source NFV Management and Orchestration (MANO) software stack aligned with ETSI NFV. `https://osm.etsi.org`
ONAP	Harmonized and comprehensive framework for real-time, policy-driven software automation of virtual network functions that enables the rapid creation of new services. `https://www.onap.org`
Open Baton	An open source reference implementation of the ETSI Network Function Virtualization MANO specification. `https://openbaton.github.io`
⋮	⋮

⋮	⋮
Open-O	Enables telecommunications and cable operators to effectively deliver end-to-end services across NFVI, as well as SDN and legacy network services. https://www.open-o.org
Cloudify	GigaSpaces' full application lifecycle orchestration tool based on based on TOSCA (Topology and Orchestration Specification for Cloud Applications) http://getcloudify.org
Tacker	OpenStack project that implements a NFV Orchestrator service with a built-in general purpose VNF Manager. https://github.com/openstack/tacker

B.2. Control Plane

Name	Description
OpenStack	Free and open-source software platform for cloud computing, mostly deployed as an IaaS, consisting of interrelated components that control diverse, multi-vendor hardware pools of processing, storage, and networking resources throughout a data center. https://www.openstack.org
OpenDaylight	The largest open source SDN controller. https://www.opendaylight.org
ONOS	A carrier-grade SDN network operating system designed for high availability, scalability, high performance and abstractions to make it easy to create apps and services. http://onosproject.org
OpenContrail	An open-source network virtualization platform for the cloud. http://www.opencontrail.org
Neutron	OpenStack project that enables network connectivity as a service for other OpenStack services. https://github.com/openstack/neutron
Keystone	OpenStack project that provides authentication, authorization and service discovery mechanisms. https://github.com/openstack/keystone
Ceph	Unified, distributed storage system designed for excellent performance, reliability and scalability. http://ceph.com
Swift	OpenStack project implementing a distributed object storage system designed to scale from a single machine to thousands of servers. https://github.com/openstack/swift
Nova	OpenStack project that provides a cloud computing fabric controller, supporting a wide variety of compute technologies. https://github.com/openstack/nova

B.3. Data Plane

Name	Description
FD.io	A collection of several projects and libraries to amplify the transformation to support flexible, programmable and composable services on a generic hardware platform. https://fd.io
Open vSwitch	A production quality, multilayer virtual switch designed to enable massive network automation. http://openvswitch.org
DPDK	The Data Plane Development Kit is a set of libraries and drivers for fast packet processing designed to run on any processor architecture. http://dpdk.org
OpenDataPlane	An open-source, cross-platform set of APIs for the networking data plane. https://www.opendataplane.org

B.4. Virtualized Infrastructure Management

Name	Description
OpenVIM	Lightweight implementation of an NFV VIM supporting Enhanced Platform Awareness features and control of an underlay switching infrastructure through an OpenFlow Controller. Now integrated into Open Source MANO (OSM).
OpenNebula	Simple yet powerful turnkey solution to build clouds and manage data center virtualization. https://opennebula.org
Citrix XenServer	A leading virtualization management platform optimized for application, desktop and server virtualization infrastructures. https://www.citrix.com/products/xenserver
⋮	⋮

⋮	⋮
VMware vCloud Director	It orchestrates the provisioning of public and hybrid cloud services to deliver complete virtual data centers for easy consumption in minutes. http://www.vmware.com/products/vcloud-director.html
Kubernetes	An open source, production-grade container orchestration system originally designed by Google and donated to the Cloud Native Computing Foundation. https://kubernetes.io
Docker Swarm	A clustering system for Docker container orchestration. https://github.com/docker/swarm
Nomad	A distributed, highly available, datacenter-aware cluster manager and scheduler. https://www.nomadproject.io
Mesosphere Enterprise DC/OS	Commercial container orchestrator based on Apache Mesos, an open-source distributed systems kernel. https://mesosphere.com

B.5. Hypervisors

Name	Description
KVM	Kernel-based Virtual Machine is a virtualization infrastructure for the Linux kernel that turns it into a hypervisor. https://www.linux-kvm.org
Xen	Leading open source virtualization platform that is powering some of the largest clouds in production today. https://www.xenproject.org
QEMU	The Quick Emulator is a generic and open source machine emulator and virtualizer. http://www.qemu-project.org
VirtualBox	Oracle's powerful x86 and AMD64/Intel64 virtualization product for enterprise as well as home use. https://www.virtualbox.org
Hyper-V	Microsoft's Hypervisor that can create virtual machines on x86-64 systems running Windows. https://www.microsoft.com/en-us/cloud-platform/server-virtualization
ESXi	VMware's enterprise-class bare-metal hypervisor for deploying and serving virtual computers. http://www.vmware.com/products/esxi-and-esx.html
LXD	A container hypervisor and a new user experience for Linux Containers (LXC). https://linuxcontainers.org/lxd

B.6. Software Containers

Name	Description
chroot	Available in most UNIX-like operating systems since 1982.
LXC	Set of tools, templates, library and language bindings related to the userspace interface for the Linux kernel containment features. https://linuxcontainers.org
Docker	World's leading software container platform. https://www.docker.com
Ubuntu Snaps	Packages any application for every Linux desktop, server, cloud or device. https://snapcraft.io
Virtuozzo	Commercial platform for secure, high performance virtualized services. https://virtuozzo.com

B.7. Software Defined GNSS receivers

Name	Description
SX3	Complete commercial package including one RF front-end with single- or dual-RF input, notebook PC and advanced navigation software. Developed by IFEN GmbH. http://www.ifen.com
ARAMIS	GNSS software-defined receiver developed by IP-Solutions and JAXA, and available in Academic, Professional and Development versions. http://www.ip-solutions.jp/SDR_GNSS_Receiver.html
Piksi	Software-based RTK GNSS modules developed by Swift Navigation. https://www.swiftnav.com
⋮	⋮

⋮	⋮
Nusar	The Navigation User SoftwAre Receiver is a flexible platform developed by GMV. http://www.gmv.com/en/Products/nusar
GSN	Software-defined receiver solution developed by Galileo Satellite Navigation Ltd. http://galileo-nav.com
GSNRx	The GNSS Software Navigation Receiver software suite is a collection of C++ class-based GNSS software receiver programs developed by the Position, Location And Navigation (PLAN) Group in the Department of Geomatics Engineering at the University of Calgary http://plan.geomatics.ucalgary.ca/software_gsnrx.php
TRIGR	Transform-Domain Instrumentation GPS Receiver developed at the University of Ohio and sponsored by the US Federal Aviation Administration. See [5].
NAMURU	Navigational Apparatus Made at UNSW for Reconfiguration by Users is a software defined GPS receiver developed at the University of New South Wales, Australia, in cooperation with General Dynamics, New Zealand. See [6].
N-GENE	A GNSS receiver completely designed in software able to run in real-time on a common Personal Computer. Developed at ISMB and Politecnco di Torino, see [7].
OpenSource GPS	Software for x86 PCs that allows you to acquire, track and demodulate signals from GPS satellites. https://sourceforge.net/projects/osgps
GPS-SDR	Open source GPS receiver based on USRP and GN3S. http://github.com/gps-sdr/gps-sdr
SoftGNSS	Matlab-based GNSS receiver, accompanying software of [14]. http://gfix.dk/matlab-gnss-sdr-book
GNSS-SDRLIB	An Open Source GNSS Software Defined Radio Library. See [15]. https://github.com/taroz/GNSS-SDRLIB
GNSS-SDR	The open source GNSS software-defined receiver used in this paper. http://gnss-sdr.org

B.8. Virtual GNSS receivers

Name	Description
GNSS-SDR Docker	https://github.com/carlesfernandez/docker-gnsssdr
GNSS-SDR Snap	https://github.com/carlesfernandez/snapcraft-sandbox

ACKNOWLEDGEMENTS

This work has been supported by the Spanish Ministry of Economy and Competitiveness through a joint collaboration of projects TEC2015-69868-C2-2-R (ADVENTURE) and TEC2015-69256-R (DESTELLO).

REFERENCES

[1] A. Checko, H. L. Christiansen, Y. Yan, L. Scolari, G. Kardaras, M. S. Berger, and L. Dittmann, "Cloud RAN for mobile networks - A technology overview," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 1, pp. 405–426, June 2015, DOI: 10.1109/COMST.2014.2355255.

[2] B. Cohen, "PaaS: New opportunities for cloud application development," *Computer*, vol. 46, no. 9, pp. 97–100, Sep. 2013, DOI: 10.1109/MC.2013.323.

[3] N. Serrano, G. Gallardo, and J. Hernantes, "Infrastructure as a Service and cloud technologies," *IEEE Software*, vol. 32, no. 2, pp. 30–36, Sep. 2015, DOI: 10.1109/MS.2015.43.

[4] M. Pathan, R. K. Sitaraman, and D. Robinson, Eds., *Advanced Content Delivery, Streaming, and Cloud Services*, John Wiley & Sons, Hoboken, NJ, Sep. 2014, ISBN: 978-0-521-76054-6.

[5] A. Soloviev, S. Gunawardena, and F. van Graas, "Development of high performance high update rate reference GPS receiver," in *Proc. of the 18th Intl. Technical Meeting of the Satellite Division of The Institute of Navigation*, Long Beach, CA, Sept. 2005, pp. 1621–1631.

[6] P. J. Mumford, K. Parkinson, and A. Dempster, "The Namuru open GNSS research receiver," in *Proc. of the 19th Intl. Technical Meeting of the Satellite Division of The Institute of Navigation*, Fort Worth, TX, Sept. 2006, pp. 2847–2855.

[7] M. Fantino, A. Molino, and M. Nicola, "NGene GNSS receiver: Benefits of software radio in navigation," in *Proc. of the European Navigation Conf. on Global Navigation Satellite Systems*, Naples, Italy, May 2009.

[8] PLAN Group, University of Calgary, "GSNRx^{TM} - GNSS Software Navigation Receiver," Tech. Rep., Calgary, Canada, Nov. 2012.

[9] IFEN Gmbh, "SX3 Black Edition Datasheet," Tech. Rep., Poing, Germany, June 2015.

[10] I. G. Petrovski and T. Tsujii, Eds., *Digital Satellite Navigation and Geophysics*, Cambridge University Press, Cambridge, UK, Mar. 2012, ISBN: 978-1-118-57521-5.

[11] Swift Navigation, Inc., "Piksi Datasheet. Flexible, high-performance GPS receiver platform running open-source software," Tech. Rep., San Francisco, CA, Mar. 2016.

[12] C. Kelley, "OpenSource GPS open source software for learning about GPS," in *Proc. of the 18th Intl. Technical Meeting of the Satellite Division of The Institute of Navigation*, Long Beach, CA, Sep. 2005, pp. 2800–2810.

[13] G. W. Heckler and J. L. Garrison, "SIMD correlator library for GNSS software receivers," *GPS Solutions*, vol. 10, no. 4, pp. 269–276, Nov. 2006, DOI: 10.1007/s10291-006-0037-5.

[14] K. Borre, D. M. Akos, N. Bertelsen, P. Rinder, and S. H. Jensen, *A Software-Defined GPS and Galileo Receiver - A Single-Frequency Approach*, Birkhäuser, New York, NY, 2007, ISBN: 978-0-8176-4390-4.

[15] T. Suzuki and N. Kubo, "GNSS-SDRLIB: An open-source and real-time GNSS software defined radio library," in *Proc. of the 27th Intl. Technical Meeting of the Satellite Division of The Institute of Navigation*, Tampa, FL, Sept. 2014, pp. 1364–1375.

[16] C. Fernández–Prades, J. Arribas, P. Closas, C. Avilés, and L. Esteve, "GNSS-SDR: An open source tool for researchers and developers," in *Proc. of the 24th Intl. Technical Meeting of the Satellite Division of The Institute of Navigation*, Portland, OR, Sept. 2011, pp. 780–794.

[17] Y. Qian, X. Cui, M. Lu, and Z. Feng, "Snapshot positioning for unaided GPS software receivers," in *Proc. of the 21st Intl. Technical Meeting of the Satellite Division of The Institute of Navigation*, Savannah, GA, Sep. 2008, pp. 2343–2350.

[18] B. Wales, L. Tarazona, and M. Bavaro, "Snapshot positioning for low-power miniaturised spaceborne GNSS receivers," in *Proc. of the 5th Satellite Navigation Technologies and European Workshop on GNSS Signals and Signal Processing*, Noordwijk, The Netherlands, Dec. 2010, DOI: 10.1109/NAVITEC.2010.5708033.

[19] I. Fernández-Hernández and K. Borre, "Snapshot positioning without initial information," *GPS Solutions*, vol. 10, no. 4, pp. 605–616, Oct. 2016, DOI: 10.1007/s10291-016-0530-4.

[20] J. Liu, B. Priyantha, T. Hart, H. S. Ramos, A. A. F. Loureiro, and Q. Wang, "Snapshot positioning for unaided GPS software receivers," in *Proc. of the 10th ACM Conference on Embedded Networked Sensor Systems*, Toronto, ON, Nov. 2012, pp. 1–14.

[21] J. Liu, B. Priyantha, T. Hart, Y. Jin, W. Lee, V. Raghunathan, H. S. Ramos, and Q. Wang, "CO-GPS: Energy efficient GPS sensing with cloud offloading," *IEEE Transactions on Mobile Computing*, vol. 15, no. 6, pp. 1348–1361, June 2016, DOI: 10.1109/TMC.2015.2446461.

[22] V. Lucas-Sabola, G. Seco-Granados, J. A. López-Salcedo, J. A. García-Molina, and M. Crisci, "Demonstration of cloud GNSS signal processing," in *Proc. of the 29th Intl. Technical Meeting of the Satellite Division of The Institute of Navigation*, Portland, OR, Sep. 2016, pp. 34–43.

[23] V. Lucas-Sabola, G. Seco-Granados, J. A. López-Salcedo, J. A. García-Molina, and M. Crisci, "Computational performance of a cloud GNSS receiver using multi-thread parallelization," in *Proc. of the 8th Satellite Navigation Technologies and European Workshop on GNSS Signals and Signal Processing*, Noordwijk, The Netherlands, Dec. 2016, DOI: 10.1109/NAVITEC.2016.7849357.

[24] ETSI ISG NFV, "NFV Release 2 Description, v1.1," Tech. Rep., ETSI, Mar. 2017.

[25] ETSI ISG NFV, "NFV Release 3 Description, v0.5.0," Tech. Rep., ETSI, Mar. 2017.

[26] R. Casellas, R. Muñoz, R. Vilalta, and R. Martínez, "Orchestration of IT/cloud and networks: From Inter-DC interconnection to SDN/NFV 5G services," in *Proc. of the Intl. Conf. on Optical Network Design and Modeling (ONDM)*, Cartagena, Spain, May 2016, pp. 1–6, DOI: 10.1109/ONDM.2016.7494060.

[27] ETSI, "OSM Release One. An ETSI OSM Community White Paper," Tech. Rep., Sophia Antipolis Cedex, France, Oct. 2016.

[28] ETSI ISG NFV, "Network Functions Virtualisation (NFV); Management and Orchestration. V1.1.1. Ref: ETSI GS NFV-MAN 001," Tech. Rep., Sophia Antipolis Cedex, France, Dec. 2014.

[29] Open Networking Foundation, "OpenFlow Switch Specification, v1.5.0," Tech. Rep., Dec. 2014.

[30] H. Al Raweshidy and S. Komaki, Eds., *Radio Over Fiber Technologies for Mobile Communications Networks*, Artech House, Norwood, MA, 2002, ISBN: 978-1-580-53148-1.

[31] D. Wake, A. Nkansah, and J. Gomes, "Radio over fiber link design for next generation wireless systems," *Journal of Lightwave Technology*, vol. 28, no. 16, pp. 2456–2464, Aug. 2010, DOI: 10.1109/JLT.2010.2045103.

[32] D. Novak, R. B. Waterhouse, A. Nirmalathas, C. Lim, P. A. Gamage, T. R. Clark Jr., M. L. Dennis, and J. A. Nanzer, "Radio over fiber link design for next generation wireless systems," *IEEE Journal of Quantum Electronics*, vol. 52, no. 1, pp. 1–11, Jan. 2016, DOI: 10.1109/JQE.2015.2504107.

[33] Ofcom, "Measuring mobile broadband performance in the UK. 4G and 3G network performance," Tech. Rep., London, UK, Nov. 2014.

[34] V. W. S. Wong, R. Schober, D. W. K. Ng, and S.-C. Wang, Eds., *Key Technologies for 5G Wireless Systems*, Cambridge University Press, Cambridge, UK, May 2017, ISBN: 978-1-1071-7241-8.

[35] H. Hurskainen, E.-S. Lohan, J. Nurmi, S. Sand, C. Mensing, and M. Detratti, "Optimal dual frequency combination for Galileo mass market receiver baseband," in *Proc. of IEEE Workshop on Signal Processing Systems*, Tampere, Finland, Oct. 2009, pp. 261–266, DOI: 10.1109/SIPS.2009.5336262.

[36] A. Farrel and I. Bryskin, *GMPLS: Architecture and Applications*, Morgan Kaufmann Publishers, San Francisco, CA, 2006.

[37] W. Felter, A. Ferreira, R. Rajamony, and J. Rubio, "An updated performance comparison of virtual machines and Linux containers," in *Proc. of the IEEE Intl. Symp. on Performance Analysis of Systems and Software*, Philadelphia, PA, Mar. 2015, pp. 171–172, DOI: 10.1109/ISPASS.2015.7095802.

[38] J. F. Zumberge, M. B. Heflin, D. C. Jefferson, M. M. Watkins, and F. H. Webb, "Precise point positioning for the efficient and robust analysis of GPS data from large networks," *Journal of Geophysical Research: Solid Earth*, vol. 102, no. B3, pp. 5005–5017, Mar. 1997.

[39] A. Rovira-García, *Consolidation and assessment of a technique to provide Fast and Precise Point Positioning (Fast-PPP)*, Ph.D. thesis, Doctoral Program in Aerospace Science & Technology, Universitat Politècnica de Catalunya, Barcelona, Spain, Jan. 2016.

[40] L. Wanninger, "Ionospheric disturbance indices for RTK and network RTK positioning," in *Proc. of the 17th Intl. Technical Meeting of the Satellite Division of The Institute of Navigation*, Long Beach, CA, Sep. 2004, pp. 2849–2854.

[41] M. Hernández Pajares, J. M. Juan, J. Sanz, A. Aragon Angel, P. Ramos Bosch, D. Odijk, P. F. de Bakker, H. van der Marel, S. Verhagen, I. Fernández Hernández, M. Toledo, and J. Samson, "Feasibility study of a European Wide Area Real Time Kinematic system," in *Proc. of 4th ESA Workshop on Satellite Navigation User Equipment Technologies*, Noordwijk, The Netherlands, Dec. 2008, pp. 1–10.

[42] C. Fernández–Prades, J. Arribas, and P. Closas, "Robust GNSS receivers by array signal processing: Theory and implementation," *Proceedings of the IEEE*, vol. 104, no. 6, pp. 1207–1220, Mar. 2016, DOI: 10.1109/JPROC.2016.2532963.

[43] J. Arribas, C. Fernández–Prades, and P. Closas, "GESTALT: A testbed for experimentation and validation of GNSS software receivers," in *Proc. of the 18th Intl. Technical Meeting of the Satellite Division of The Institute of Navigation*, Tampa, FL, Sep. 2015, pp. 3222–3234.

[44] Fundació Centre Tecnològic de Telecomunicacions de Catalunya, "ADRENALINE Testbed® website," `http://networks.cttc.es/ons/adrenaline/`, 2017, [Online; accessed September 27, 2017].

[45] R. Vilalta, A. Mayoral, R. Muñoz, R. Casellas, and R. Martínez, "The SDN/NFV cloud computing platform and transport network of the ADRENALINE testbed," in *Proc. of the IEEE Conf. on Network Softwarization*, London, UK, Apr. 2015, pp. 1–5, DOI: 10.1109/NETSOFT.2015.7116150.