*future internet*

MDPI

*Article*

# The Next Generation Platform as A Service: Composition and Deployment of Platforms and Services

**Angelos Mimidis-Kentis [1,*], Jose Soler [1], Paul Veitch [2], Adam Broadbent [2], Marco Mobilio [3], Oliviero Riganelli [3], Steven Van Rossem [4], Wouter Tavernier [4] and Bessem Sayadi [5]**

[1] Department of Photonics, Technical University of Denmark, 2800 Lyngby, Denmark; joss@fotonik.dtu.dk
[2] British Telecom, Ipswich IP53RE, UK; paul.veitch@bt.com (P.V.); adam.broadbent@bt.com (A.B.)
[3] Department of Informatics, University of Milano Bicocca, 20125 Milano, Italy; marco.mobilio@disco.unimib.it (M.M.); oliviero.riganelli@unimib.it (O.R.)
[4] IMEC, University of Ghent, 9000 Ghent, Belgium; steven.vanrossem@ugent.be (S.V.R); wouter.tavernier@ugent.be (W.T.)
[5] Nokia Bell Labs France, 91620 Nozay, France; bessem.sayadi@nokia-bell-labs.com
* Correspondence: agmimi@fotonik.dtu.dk

Received: 9 April 2019; Accepted: 17 May 2019; Published: 21 May 2019

**Abstract:** The emergence of widespread cloudification and virtualisation promises increased flexibility, scalability, and programmability for the deployment of services by Vertical Service Providers (VSPs). This cloudification also improves service and network management, reducing the Capital and Operational Expenses (CAPEX, OPEX). A truly cloud-native approach is essential, since 5G will provide a diverse range of services - many requiring stringent performance guarantees while maximising flexibility and agility despite the technological diversity. This paper proposes a workflow based on the principles of build-to-order, Build-Ship-Run, and automation; following the Next Generation Platform as a Service (NGPaaS) vision. Through the concept of Reusable Functional Blocks (RFBs), an enhancement to Virtual Network Functions, this methodology allows a VSP to deploy and manage platforms and services, agnostic to the underlying technologies, protocols, and APIs. To validate the proposed workflow, a use case is also presented herein, which illustrates both the deployment of the underlying platform by the Telco operator and of the services that run on top of it. In this use case, the NGPaaS operator facilitates a VSP to provide Virtual Network Function as a Service (VNFaaS) capabilities for its end customers.

**Keywords:** 5G; Cloud; Cloud-native; PaaS; NFV; SDN; Telco-grade; CORD

## 1. Introduction

5G is expected to be the ubiquitous fabric to provide reliable, low-latency and high-speed connectivity for a wide variety of services (Internet of Things (IoT), mobile, Industry 4.0, automotive, etc.). However current network architectures are not yet able to provide these characteristics, mainly due to their monolithic nature which limits their flexibility, scalability, and programmability. To facilitate the adoption of 5G, the consensus in the industry is to utilise technologies and workflows from the field of Information Technology (IT), such as the cloudification and virtualisation of services [1]. This way, network services that previously comprised dedicated and monolithic hardware appliances (e.g., Firewalls) will now be virtualised instances in commodity servers in the cloud. This shift is expected to offer a much higher degree of flexibility, scalability, and automation to service providers, as they will be able to modify the virtual setups dynamically and programmatically.

To fully reap the benefits provided by virtualisation and cloudification, the deployment and management of services must be decoupled from the underlying computing and networking infrastructure. A paradigm that can facilitate this requirement is the Platform as a Service (PaaS), as it provides a centralised and reliable environment, with high degrees of agility and automation, on which services can be deployed and managed. The benefits provided by the PaaS model can be further enhanced by incorporating the paradigms of Software Defined Networking (SDN) and Network Function Virtualisation (NFV). The scope of SDN would be to facilitate the control and programmability of the network infrastructure while NFV would allow for the efficient virtualisation and management of services on commodity servers.

In contrast with IT services, 5G services are more diverse and stringent in terms of their network and computing requirements. These requirements imply the need for a more flexible and customisable PaaS model [2] than the currently available non 5G oriented solutions. One such model is the Next Generation Platform as a Service (NGPaaS) [3,4], which is built around the principles of micro-services, modularity, and build-to-order. In short, NGPaaS allows for the deployment of custom-built platforms and services, on a diverse set of infrastructure technologies, using automated and technology agnostic workflows. Through the same workflow, any Vertical Service Provider (VSP) (e.g., a Telco provider or IoT provider) can deploy either a custom platform on the available infrastructure or services on top of an existing platform.

The content of this paper is structured as follows. At first, a study of related work is presented in Section 2. Section 3 presents the implementation of the requirements and architecture of NGPaaS, as conceptualised in references [3,4]—specifically, it provides the NGPaaS architectural design, the concept of Reusable Functional Blocks (RFBs) and introduces the RFB Description and Composition Languages Design, Deploy and Direct (RDCL 3D) tool [5]. Later in Section 4, the NGPaaS workflow is practically validated through a use case. This use case (Telco PaaS) is based on the Central Office Re-architected as a Datacentre (CORD) [6] platform and provides Virtual Network Function (VNF) as a Service (VNFaaS) capabilities to a VSP. In addition to describing the use case and the components that comprise it, this paper also details the process of how these components were made NGPaaS-ready in Section 5. Later, in Section 6 a brief description of another use case targeted by NGPaaS (5G Public Safety) is provided. Finally, Section 7 provides a critical conclusion of the presented work.

## 2. Related Work

As mentioned in the introduction, most telco-grade platform environments have originated from existing IT-based cloud technologies (e.g., OpenStack, Kubernetes), but have been extended with new features to be suitable for telco environments, such as edge, access, and core networks. The scope of this section is to contextualise telco-grade platform solutions with recent PaaS and NFV related work.

The most prominent standardisation effort in the NFV platform area is the ETSI NFV Management and Orchestration (MANO) specification [7], which details an NFV-based platform architecture. There, a fixed set of PaaS related features is coupled tightly to the underlying infrastructure managers. However, unlike NGPaaS, ETSI NFV MANO lacks the option to customise which specific PaaS components are included in a deployment, consequently, the supported virtualisation technologies and infrastructure types cannot be modified easily or swapped. Instead, the MANO functionality focuses on the lifecycle management of specific services, controlled by pre-integrated infrastructure managers and PaaS functions. Different flavours of MANO platforms are available (e.g., OSM [8] or 5GTANGO [9]), and each one may have proprietary interfaces and service descriptor formats, which can act as a barrier to the integration of other PaaS solutions. The authors in reference [10] and reference [11] show that the characteristics might differ significantly when comparing existing MANO solutions. It may be necessary, depending on the targeted use-case, to augment an existing MANO platform and integrate new functionalities ([10,12]).

In addition to MANO platforms, there are many advances are being made in the NFV technology space. One such example is that in addition to common x86 server hardware, the use of FPGAs in data centre infrastructure is being presented as virtualised resources, as demonstrated in reference [13]. In reference [14], a management framework is presented for using FPGA resources in data centres, also enabling FPGAs to run Virtual Network Functions (VNFs). The authors of reference [15] have implemented a framework to design network functions running on split CPU and FPGA architectures—this provides a method to offload computationally–intensive processing of the network functions to the FPGA.

The development of improved algorithms to assist in the deployment and operation of virtualised network functions is a highly active research area. A method for customised VNF placement is proposed in 5G networks [16], showing that due to the diverse performance requirements among different 5G scenarios, an adaptive VNF placement approach is needed to accommodate service-specific requirements automatically. In addition to placement, scaling algorithms can also enhance the operation of VNFs. In reference [17], a formal method is described to predict VNF traffic and optimally scale the VNF resources accordingly in an elastic way. Similarly, in reference [18], a novel analytical model based on Stochastic Network Calculus is presented to investigate the end-to-end performance bound of chained VNFs quantitatively. By modelling (non-)bursty network traffic, the proposed analytical model provides an efficient method for service providers to determine which service chaining and placement configuration are more beneficial to meet the SLA requirements in terms of violation error and latency performance. Another operational enhancement is an optimised load balancing system for VNFs, presented in reference [19] and the placement of multiple VNFs in geo-distributed infrastructures is investigated in reference [20]. The latter is particularly applicable in a telecom scenario, where an optimal choice must be made in which central offices the VNFs should run.

In additional to VNF placement and scaling, automated healing actions also need to be implemented to quickly analyse and solve any issues with the running network service before performance is degraded. Autonomous healing actions in reference [21] are based on pre-defined workflows developed by experts. The primary idea behind the adopted approach is to transfer the current expert domain knowledge and manual action triggers into a rule-based self-healing solution. Moreover, in reference [22], it is shown that deep learning can be used to identify anomaly events from NFV system logs reliably. A technique using a supervised machine learning method for online classification of anomaly states based on similarities between anomaly type-specific density grid patterns is presented in reference [23]. The detection is done by analysing CPU, memory and network usage; the procedure used, the density grid mapping, is inherited from the theory of grid-based clustering. Operational procedures such as healing combine techniques from various research areas and gather information from a wide range of data, implying the integration of a diverse set of tools and libraries. The above-described references show that the PaaS should cater for quick and easy upgradeability, in order to incorporate the rapid evolutions happening in the NFV area and to allow easy customisation for a Telco or other use-case. The platform should follow a modular approach, and a relevant PaaS architecture for this method is advised in reference [3]. Later, in Section 4, we exemplify this modular setup by showing how we implemented in NGPaaS the functional blocks of the orchestration mechanism, network control, monitoring framework and the analysis of VNF anomalies to trigger healing actions.

According to reference [24], a unified PaaS interface can be derived from existing PaaS Application Programming Interfaces (APIs), or put another way, a common set of functionalities regarding service instantiation and configuration can be grouped under a standard API. This allows for a unified interface for application deployment and management, among different cloud platforms, avoiding technology lock-in effects. Unfortunately, considerable effort is needed to maintain and translate the unified API to the proprietary PaaS APIs; a common trade-off between the overhead and benefits of using an API abstraction layer. In contrast to reference [24], NGPaaS adopts a workflow-based orchestration mechanism [3], hence the API or descriptor format of the targeted PaaS or infrastructure manager is addressed natively in the workflow and thus minimises the effort required to integrate support for new APIs.

The NGPaaS architecture is innovative in the sense that both PaaS and service components are considered as the same module type, using the same generic orchestration mechanism based on the RFB model. As argued in reference [25], if software suppliers adopt the generic RFB descriptor model, DevOps-based integration strategies will be greatly facilitated. The NGPaaS framework, therefore, facilitates customisation and the creation of a custom PaaS solution, with functionality targeted at specific use cases. Given the vast range of possible services, it can be challenging to integrate all existing options into a one-size-fits-all platform solution. NGPaaS implements the complete service lifecycle using build-ship-run actions: (1) What is needed in terms of service offering (build), (2) How the service can be deployed (ship) and (3) Where the service should be deployed (run). These fundamental design questions facilitate the mapping of the right PaaS technologies to the available IaaS and the requirements of the service that should be executed. Each PaaS can be enhanced with unique features related to network state, Quality of Service, high availability, auto-scaling or SDK toolsets [25]. The deployed set of PaaS features is then customisable and use-case dependent. Before exemplifying this through the Telco PaaS use case, we first elaborate on the different processes and workflows which enable this modular NGPaaS framework.

## 3. The Next Generation PaaS: Architecture, Concepts, Processes, and Workflows

The scope of this section is to provide the reader with an understanding of the NGPaaS. More specifically this section introduces the overall NGPaaS architecture, the concept of RFBs, how RFB graphs can be composed via the RDCL 3D tool, and finally introduces the workflow through which platforms and services can be composed and deployed in NGPaaS.

### *3.1. The NGPaaS Architecture*

NGPaaS consist of a multi-layer architecture, with each layer being responsible for a specific functionality. In total, six layers are defined as: (1) The Business Registration Layer, (2) The Business as a Service (BaaS) Layer 3) The Business and Operation Support System (BSS/OSS) Layer, (4) The Dev-for-operations Layer, (5) The Platform as a Service (PaaS) Layer and finally (6) The Infrastructure as a Service (IaaS) Layer. The details and scope of each of these layers can be found in Table 1, while Figure 1 provides a simplified view of the NGPaaS architecture and also of the cross-layer interactions.

**Table 1.** Layers of the NGPaaS architecture.

| NGPaaS Layer | Description |
|---|---|
| Business Registration | Registers all stakeholders that participate in NGPaaS (VSPs, Vendors, etc.). It also includes the resolution of access and execution rights. |
| BaaS | A customisable catalogue from which to order service or platform workloads. |
| BSS/OSS | Responsible for inventory registration, global supervision, and deployment of services and platforms on their execution environments. In our prototype, the BSS/OSS role [26] is fulfilled by the RDCL 3D tool. |
| Dev-for-Operations | An environment to support the innovative NGPaaS Dev-for-Operations model—an evolution of the existing DevOps model to facilitate new types of interaction and development methods between multiple stakeholders. This environment is where staging and development are performed to bring new PaaS or service components into NGPaaS [27]. |
| PaaS | PaaS components are deployed on the available (declared) infrastructure. These PaaS components form the framework to manage the VNFs included in the requested services. More than one PaaS instances can be active and managed at the same time by the OSS/BSS. |
| IaaS | This layer relates to the cloud infrastructure available to the NGPaaS operator. |

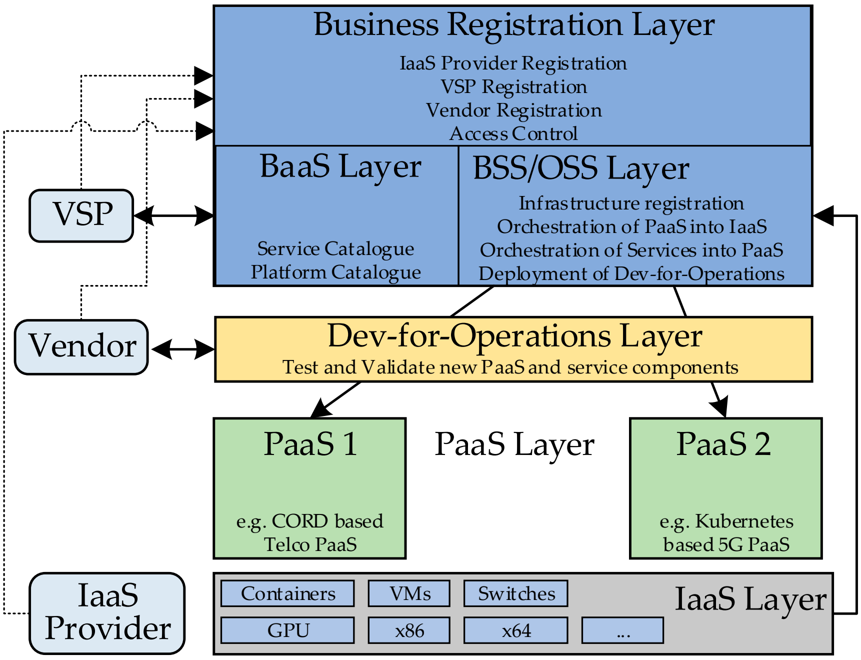


**Figure 1.** Simplified view of the NGPaaS architecture.

### *3.2. Reusable Functional Blocks*

RFBs are a core concept of NGPaaS, as they have a significant influence in its architecture and workflows. RFBs are a logical representation of functions that can decompose a complex system into simple sub-functions. In a sense, RFBs are a generalisation of the Virtual Network Function (VNF) concept proposed by ETSI [28]. There are however some core differences between ETSI VNFs and NGPaaS RFBs. The main difference is that RFBs are recursive and thus can be arbitrarily decomposed into other RFBs, while VNFs in the ETSI model cannot be decomposed into other VNFs (the composition stops at the level of VNF Component). In addition, RFBs can be mapped into both software and hardware execution environments [28], while VNFs can only be mapped into Virtual Machines (or Containers) in traditional cloud infrastructures. RFBs can also be paired with metadata during their composition, which allows for high degrees of configuration, with reduced complexity. Figure 2 illustrates the RFB concept. Additionally, NGPaaS is a higher-level orchestration framework when compared to ETSI NFV, as it facilitates the deployment and management of multiple heterogeneous platforms. For example, it could be the case that an Open Source MANO (OSM) platform, a reference implementation of ETSI NFV, is deployed and managed by NGPaaS. A detailed comparison between the ETSI NFV and NGPaaS proposal is also provided in reference [3].

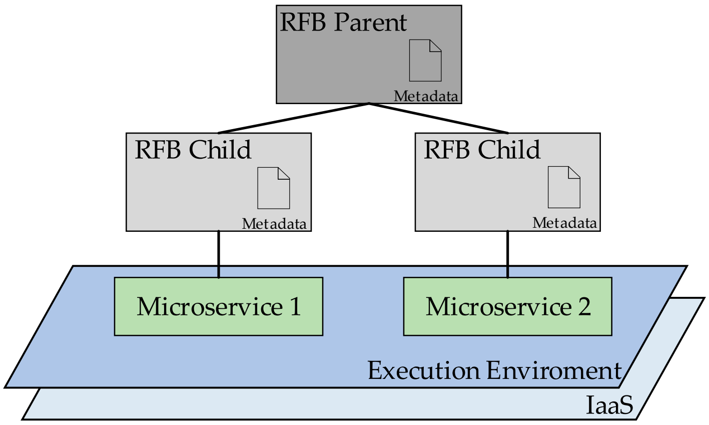


**Figure 2.** The concept of Reusable Functional Blocks.

The figure shows how an RFB parent is decomposed into two RFB children, each of which is mapped to the deployment of a micro-service into the execution environment. For these RFBs to be properly configured, they are each paired with some metadata. The NGPaaS RFB model supports inheritance from an RFB parent to its children; this means that, unless overwritten by an RFB child, metadata from the parent will be propagated to all its children.

### *3.3. RFB Description and Composition Languages Design, Deploy and Direct Tool*

To fully exploit the RFB concept, NGPaaS is utilising the RDCL 3D tool: a web-based framework that allows the composition of RFB graphs. By composing graphs comprising multiple RFBs, it is possible for a VSP to define complex platforms and services for deployment. Apart from composing RFB graphs, the RDCL 3D tool is also responsible for "shipping" them to an RDCL agent. Every RDCL agent is tied to an execution environment (e.g., an infrastructure or a platform) and its role is to deploy the services/platforms composed by the graph into this execution environment. To facilitate this action, each RFB leaf in an RFB graph is mapped to an Ansible-based [29] workflow which is executed onto the execution environment of the RFB by the agent. By selecting different execution environments for different RFBs of an RFB graph, it is possible for a deployment to span multiple execution environments. For example, a platform deployment can span across a public and private cloud infrastructure. For the RFBs to be executed in the desired order, a priority mechanism is used in which each RFB is assigned a priority value. The RDCL 3D tool was initially developed in the 5G-PPP Superfluidity [30] project but has been heavily extended to meet the requirements of NGPaaS. Figure 3 illustrates the operation of the RDCL 3D tool in greater detail.

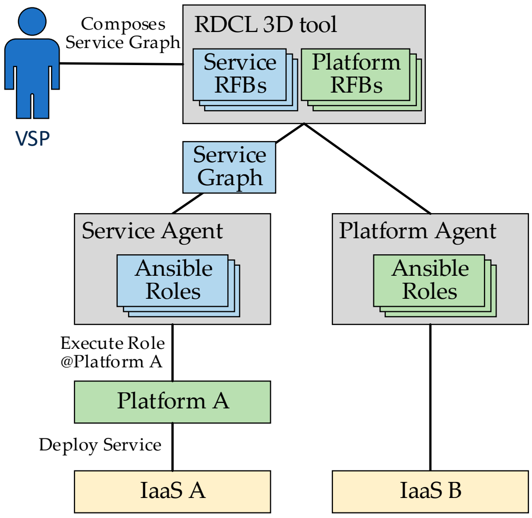
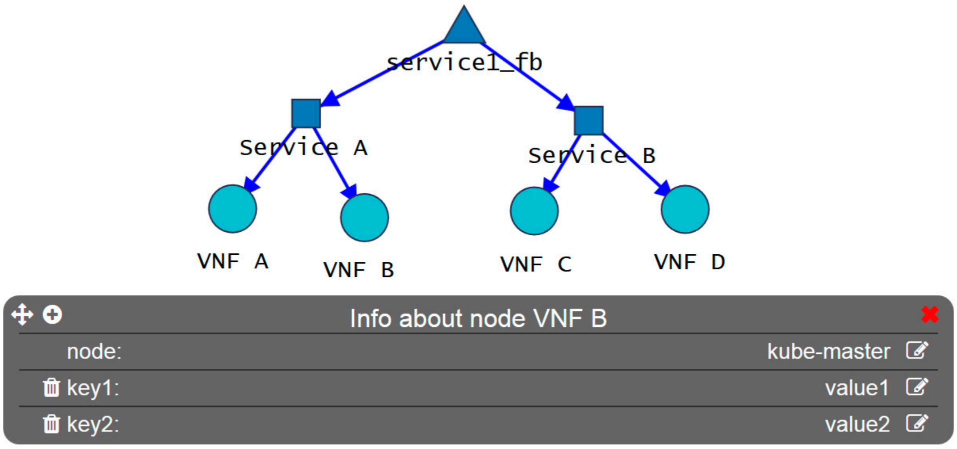


**Figure 3.** Overview of the RDCL 3D tool.

A VSP uses the web-based interface of the RDCL 3D tool to describe the desired service or platform using an RFB graph; this comprises of RFBs from the catalogue of RFBs made available by the NGPaaS operator. The RDCL 3D tool provides a full separation between platform and service deployments by assigning each RFB to either a service or platform category. Then when composing a service or platform graph, the VSP will be only presented with the corresponding RFBs. Once the composition of the RFB graph is completed, the VSP selects an appropriate RDCL agent (In this case the service agent of IaaS A). The RFB graph is encoded as a JSON string and is shipped to the agent. Once the agent receives the deployment request, it initiates the execution of the corresponding Ansible roles, each mapping to an individual RFB leaf of the RFB graph. Finally, these Ansible roles are executed into the appropriate execution environment (Platform A in this case) and the desired service is provisioned. Figure 3 also illustrates the possibility to control multiple RDCL agents, via a single RDCL 3D instance.

Figure 4 depicts a simple example of an RFB graph as it appears in the RDCL 3D tool web interface. A root RFB (service1_fb) comprises two RFB children (Service A, Service B) each corresponding to a different logical subcomponent of the parent service. Each of these RFB comprises of two RFB leaves (VNF A, VNF B, VNF C, and VNF D) -these leaf RFBs are the ones directly mapped to the deployment micro-services (e.g., containers) in the execution environment. Finally, the figure also shows as an example the metadata tied to leaf RFB VNF B (two key-value pairs).



node: kube-master

key1: value1

key2: value2

**Figure 4.** Example of RFB graph from the RDCL 3D tool.

### 3.4. Processes and Workflows

The ability to deploy, in a modular way, both the platform and service components is one of the key features of NGPaaS. This allows the creation of an eco-system where telco operators can easily cooperate and integrate with multiple software vendors. By using an aligned descriptor format between the vendor and operator side, such as the RFB model, it becomes easier to share both service components and their related execution environments (PaaS). The processes defined in this section, enables the deployment of a customisable set of PaaSes and service components, empowering the support of many use-cases. The most critical processes between layers mentioned above are illustrated in Figure 5 and also analysed in Table 2.

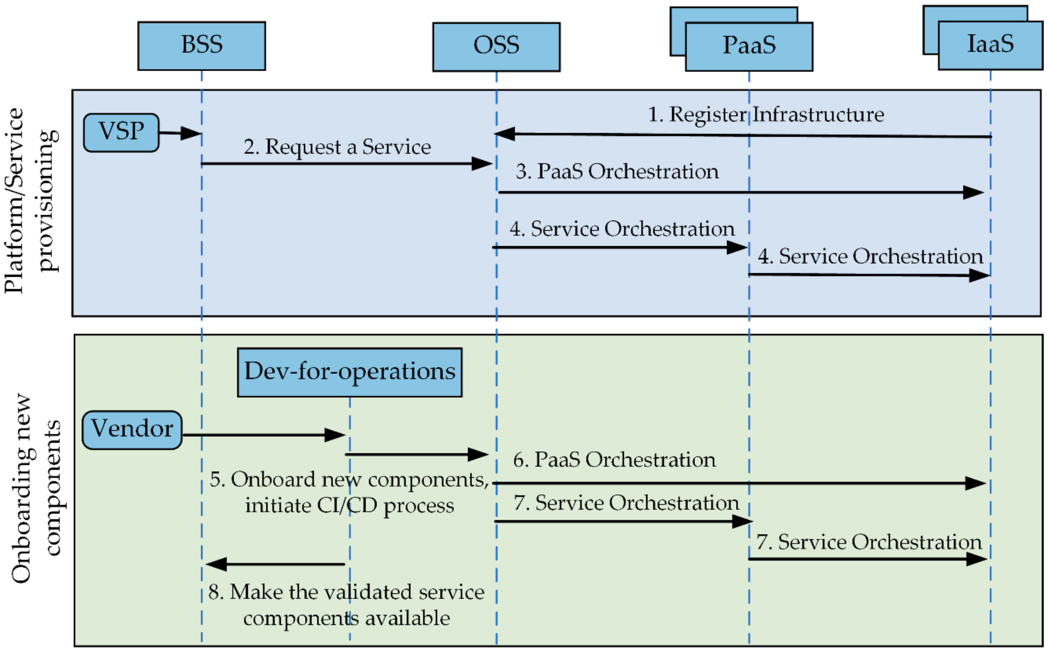


**Figure 5.** NGPaaS processes, to enable a customisable platform.

**Table 2.** Processes of the NGPaaS architecture.

| NGPaaS Process | Details |
|---|---|
| Infrastructure registration | A generic provisioning process to support the broad spectrum of available infrastructure technologies. According to the use-case and the required services, the appropriate infrastructure nodes are leased and registered in the OSS. |
| PaaS orchestration | Refers to the deployment of selected PaaS components on the appropriate IaaS. The PaaS could be Kubernetes, CORD, or any other platform. The PaaS components can be aggregated flexibly and modelled as RFBs. This capability could not be implemented using ETSI MANO, which is focused only on VNFs and assumes the platform is already deployed. |
| Service orchestration | A service is provided by deploying VNFs as sets of RFBs. The target execution environment of each VNF is included in the metadata of the related RFBs, which also specify the runtime aspect: VM, container, Unikernel, FPGA Bitstream, etc. The execution environment of a service component is a pre-deployed PaaS which supports the runtime aspects of the RFB. |
| Onboarding new components | The design rule in NGPaaS is "everything is an RFB" (VNF, ancillary services like orchestration, SDN controller, etc.). Therefore, all the components could be updated, upgraded, swapped, etc. This can be done through the usage of the Dev-for-Operations processes. |

Before requesting any platform or service deployment, (1) the required infrastructure for the VSP must be reserved by the NGPaaS operator. Once the appropriate infrastructure has been registered, then, (2) the VSP can request the deployment of the desired service. This is done by the composition of two RFB graphs, one for the underlying platform (if not already present) and one for the service itself. Once the RFB graphs have been composed then PaaS (3), or service workloads (4) can be orchestrated on their corresponding execution environments. A relatively similar workflow (5)–(8) is also available to software vendors through their dedicated Dev-for-Operations layer, which allows them (5)–(7) to test new software components that can be later (8) included as RFBs in the available RFB catalogue.

## 4. Virtual Network Function as A Service, Overview

The scope of this section is to illustrate the operation of the NGPaaS workflow, via a demonstrable use case. It outlines a service scenario called "VNF-as-a-Service" (VNFaaS), which is based on the CORD platform. More specifically, this section highlights the composition of both the platform and services as RFB graphs and their subsequent deployment in their corresponding execution environment (IaaS for CORD, PaaS for the services).

### *4.1. The Telco PaaS Use Case: Virtual Network Function as A Service (VNFaaS)*

In the Telco PaaS use case, the NGPaaS operator (e.g., a Tier 1 Telco operator) hosts VNFs on behalf of a Service Provider (the VSP). These VNFs could be virtual Routers and Firewalls, which the Service Provider can configure accordingly depending on the desired service (business VPNs, Internet access, etc.). In a "pre-NGPaaS" NFV architecture, the on-boarding of VNFs, to be made available in the service catalogue, is done by the Telco operator (based on the preferences of the Service Provider).

Using the NGPaaS workflow, the Telco operator instead utilises a PaaS-oriented approach instead. While the Telco provider still deploys the platforms, the Service Providers can onboard and administer specific VNFs on their own (e.g., vRouter, vFirewall) and also enable Value-added Service (VAS) capabilities. These VASs could be Telco-grade enhancements to the basic service (e.g., monitoring, healing, policy-based network control, etc.). This way, a pre-existing "VNF App Store" can be supplemented by allowing the Service Provider to onboard new VNFs via direct interactions with preferred vendors, resulting in a more diverse set of capabilities than could be obtainable with the pre-NGPaaS model. There many benefits associated with the adoption of the VNFaaS use case for both the VSP and its end customers. Most of these benefits are due to the nature of the VNFaaS use case, for example, the Service Provider does not need to ship hardware to the premises of its end customer, allowing for a more flexible and scalable business and service model.

Figure 6 illustrates the VNFaaS use case in more detail. There, a VSP has deployed two service graphs through the RDCL 3D tool, each comprising of two VNFs. Deploying these two graphs has resulted in the deployment of two service chains onto the CORD platform. Both service chains comprise an interconnected virtual router and firewall and are associated with a unique end user. The figure also illustrates the possibility to onboard and deploy VNFs of different vendors, thus providing flexibility to the end user.

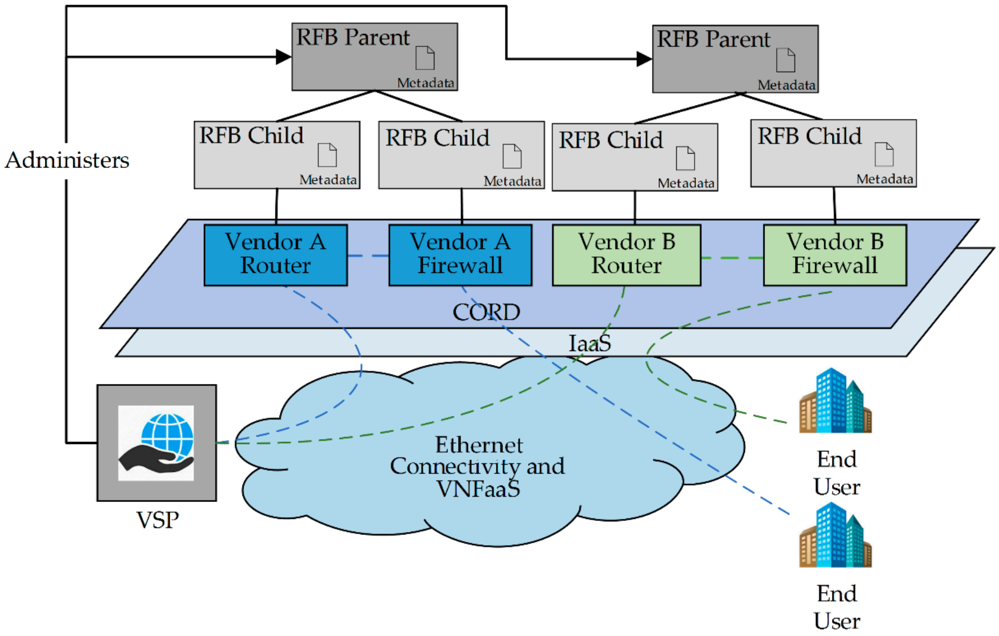


**Figure 6.** The Virtual Network Function as a Service use-case.

### 4.2. *The VNFaaS Proof of Concept*

To validate the NGPaaS workflow through the VNFaaS use case, a Proof of Concept (PoC) scenario was designed and implemented. To be considered successful, this PoC must be able to showcase both the provisioning of a Telco-grade platform and the provisioning of services related to the VNFaaS use case.

Figure 7 illustrates an overview of the desired final status of the PoC. There, a Service Provider has been granted access to the RDCL 3D tool by the NGPaaS operator. Using this tool, the Service Provider is then able to compose and deploy service graphs of virtual routers and firewalls, which will provide the desired service to the provider's end customers. Through the NGPaaS workflow, these service graphs are then translated to platform-specific APIs and passed down to the platform orchestrator. Since a service graph request will usually comprise both compute and network resources, the platform orchestrator will translate the request to API calls targeting both the underlying VIM and SDNC components of the platform. The VIM will then provision the required virtual computing resources and the SDNC will provide the required connectivity. It should also be possible for the service provider to monitor its deployment for specific KPIs. At the same time, an alerting function should continuously check the collected data and trigger automated healing workflows, when the state of a deployed service is undesirable.

The next couple of sections will provide more details on how this PoC was developed. More specifically, the technologies that comprise it will be detailed (e.g., Platform of choice), together with the steps required to integrate them into the NGPaaS workflow.

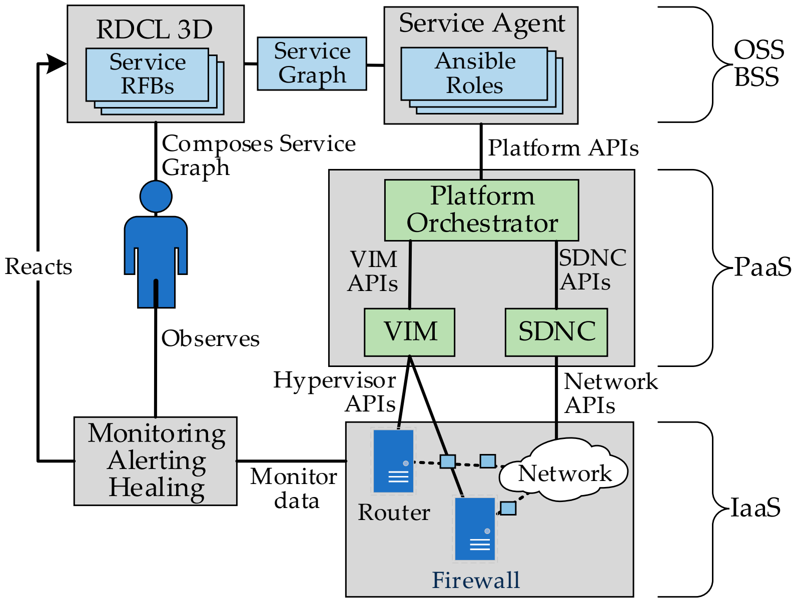


**Figure 7.** VNFaaS Proof of Concept.

### 4.3. *The Telco PaaS Platform: Central Office Re-Architected as A Datacentre (CORD)*

The selected platform for the Telco PaaS use case is CORD and the current prototype uses version 6.0 [6]. The primary reason for selecting CORD was that it was designed to provide edge network access, by acting as a virtualised Central Office. This implies that the core functionalities of the CORD platform could be effectively reused to fit the needs of the VNFaaS use case. For example, other platforms (e.g., OSM), have a more generic implementation which would make their integration into the VNFaaS use case a more complex task.

CORD's architecture comprises three main functional elements, implemented as open source projects: (1) The XOS orchestrator, responsible for the joined control of ONOS and OpenStack. This joined control is facilitated by well-defined service models, which describe the supported services and service synchronisers. The synchroniser's role is to align the operational state of CORD with the desired state, as defined by the administrator, in a programmatic way; (2) The ONOS SDN Controller (SDNC) is responsible for managing the physical and virtual networks; (3) The OpenStack platform is responsible for managing the lifecycle and resources of the deployed VNFs. In the used version of CORD, all the individual CORD components are deployed as sets of Docker containers, managed by Kubernetes and Helm. Figure 8 illustrates a high-level overview of the CORD architecture.

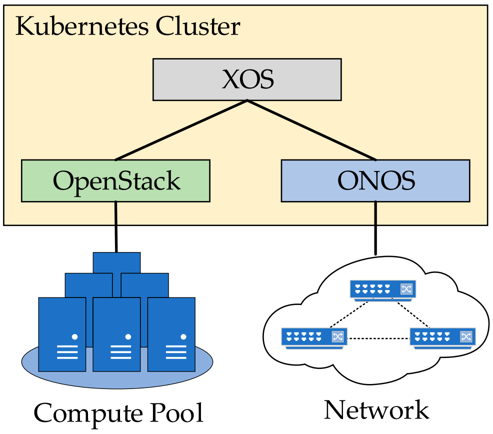


**Figure 8.** High-level overview of the CORD architecture.

### *4.4. Supported Virtual Network Functions (VNFs) and Value Added Services (VAS)*

#### 4.4.1. Core Services: Virtual Firewalls and Virtual Routers

This section gives a brief overview of the VNFs and VASs supported by the Telco PaaS prototype. As part of the experimental set-up, we use a commercially available router and firewall VNFs. The Router VNF is an enterprise-class router designed to run on x86 standard servers but comprising all the same features as equivalent hardware routers [31]. Similarly, the Firewall VNF is a virtual appliance aimed at enterprise-level security deployments [32]. The implementation of both VNFs is based on virtual machines and in the presented prototype their running instances are hosted by OpenStack.

#### 4.4.2. Value Added Services: Monitoring, Alerting, and Healing

In the Telco PaaS, monitoring utilises the ElasticSearch-Logstash-Kibana (ELK) stack [33]. This stack consists of three main components: (1) Elastic Search: A distributed search and analytics engine based on REST APIs; (2) Logstash: A server-side data processing tool that can ingest data from various probes and manipulate it before sending it to ElasticSearch; (3) Kibana: A visualisation tool provided with ElasticSearch. As noted, in the Telco PaaS prototype the selected VNFs are based on proprietary VMs, thus deploying monitoring probes internal to the VNFs is not feasible. To overcome this limitation the probes in the Telco PaaS are deployed in a side-car fashion, externally to the target service; but can poll the target service for the required using the available interfaces. For this prototype, we rely on the SNMP protocol and REST interface to monitor the router VNF and ONOS SDNC respectively. More specifically, SNMP is used to poll the virtual router for its CPU utilisation, while REST is used to poll ONOS for network traffic information regarding the virtual interfaces of the deployed VNFs.

As an enhancement to baseline monitoring, this prototype also provides Alerting and Healing capabilities. The ELK stack can map trends in the CPU utilisation of monitored VNFs to predefined anomalous behaviours. Once such an anomaly is detected, a Healing function is called, which redeploys the anomalous VNF with more resources (e.g., more virtual CPUs or RAM). All healing actions are performed, by the Healing rule via the RDCL 3D tool, thus ensuring alignment between the state of the deployment and the view of the tool. Figure 9 shows the operation of the monitoring VAS in detail. As mentioned before, monitoring, alerting, and healing are complementary services provided selectively to VSPs. As a result, upon deployment of a VNF, the VSP can set a flag in the RDCL 3D metadata to trigger or not trigger the monitoring capabilities for this VNF.

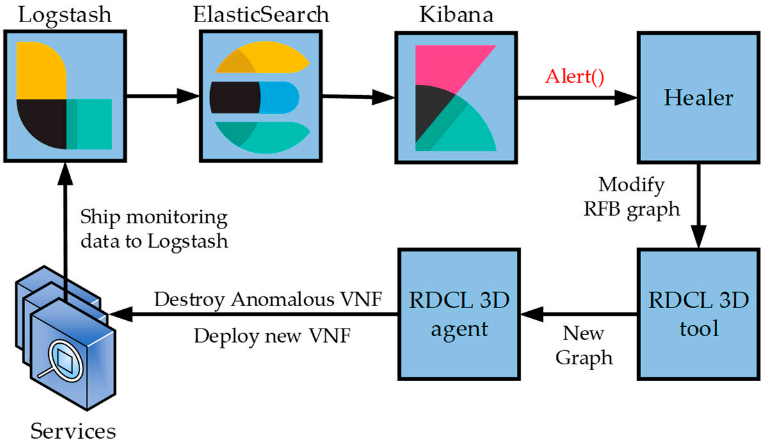
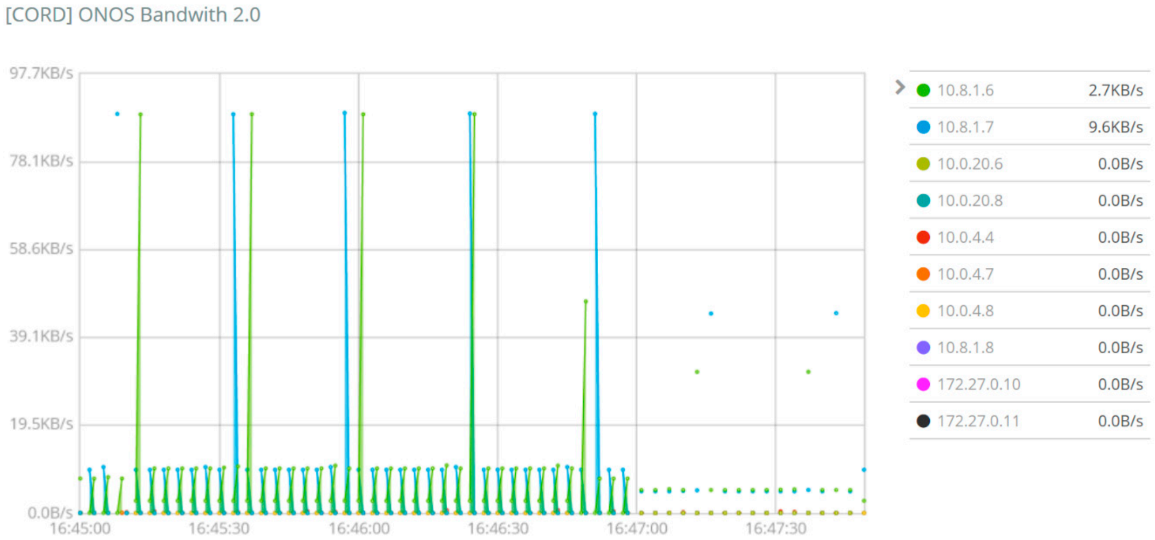


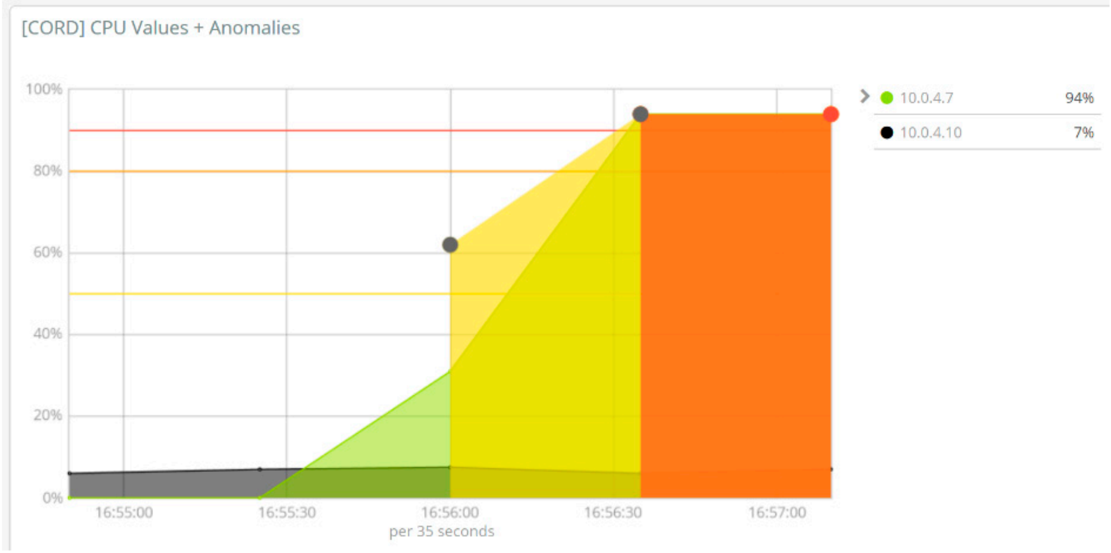
**Figure 9.** Operation of the Monitoring Value Added Service.

Figure 10 illustrates the capability for live network traffic information for each interface of the provisioned VNFs. As mentioned, this information is collected by the monitoring probe through the ONOS REST API. On the other hand, Figure 11 shows how live information is reported about the CPU utilisation of all monitored VNFs. In addition, this segment is configured with three predefined thresholds (50%, 80%, and 90%)—when any of these thresholds is reached a corresponding alert is shown to the administrator. In the provided example, the critical threshold of 90% has been exceeded (94%); exceeding this threshold triggers an automated healing workflow that will re-provision the VNF with more virtual resources.



| | |
|---|---|
| ● 10.8.1.6 | 2.7KB/s |
| ● 10.8.1.7 | 9.6KB/s |
| ● 10.0.20.6 | 0.0B/s |
| ● 10.0.20.8 | 0.0B/s |
| ● 10.0.4.4 | 0.0B/s |
| ● 10.0.4.7 | 0.0B/s |
| ● 10.0.4.8 | 0.0B/s |
| ● 10.8.1.8 | 0.0B/s |
| ● 172.27.0.10 | 0.0B/s |
| ● 172.27.0.11 | 0.0B/s |

**Figure 10.** Network Monitoring Dashboard of Telco PaaS.



| | |
|---|---|
| ● 10.0.4.7 | 94% |
| ● 10.0.4.10 | 7% |

**Figure 11.** CPU Monitoring/Healing Dashboard of Telco PaaS.

### 4.4.3. Value Added Services: Policy-Based Network Management

In addition to monitoring/alerting/healing, the Telco PaaS supports one more VAS, namely policy-based network management. This is achieved with the integration of an SDN-based network policy framework [34] to the ONOS SDNC. By acting as a layer of abstraction between the network infrastructure and the VSP, the network policy framework allows for simplified control over the network infrastructure. Also, the network policy framework follows an entirely modular architecture, consisting of a Policy Manager and multiple separate Policy Type applications, thus allowing the VSP to add support for new policy types on demand, without affecting the runtime operation of the policy framework. Figure 12 illustrates a (simplified) example operation of the policy framework. There an administrator has deployed the policy manager with two policy type implementations (Firewall and NAT). Subsequently, the administrator issues a request to create a Firewall policy instance that will drop traffic between hosts H1 and H2. Upon receiving the request, the policy manager will validate it for correctness, and if successful, it will then request the Firewall Policy Type implementation to enforce the policy in the network. This will result in the Firewall Policy Type to install flow rules in the switches adjacent to H1 and H2, which will block traffic between the two endpoints.

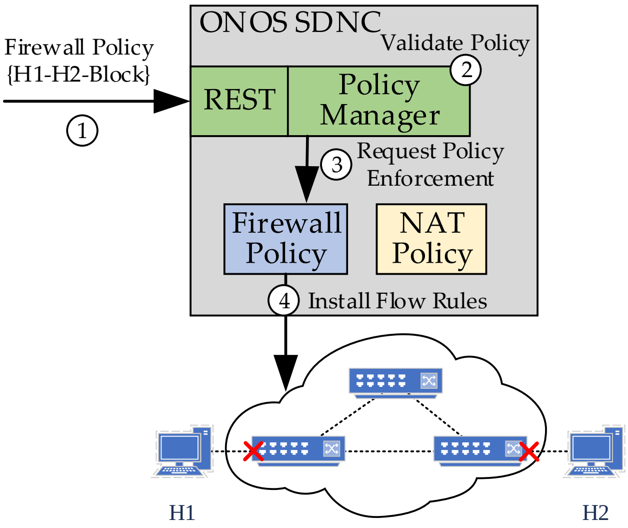


**Figure 12.** Operation of the Network Policy Framework.

## 5. Virtual Network Function as A Service: Component RFBisation

The scope of this section is to highlight how the different platform and service components that comprise the Telco PaaS use case, have been mapped to individual RFBs and RFB graphs.

### *5.1. RFBisation of the CORD Platform*

In previous versions of the CORD platform (<6.0), there was a very tight integration between the different components of CORD, since the majority of them were executed either as native services or as standalone containers and virtual machines. This translated to a very rigid deployment process, with little to no flexibility. However, in the latest version (6.0), the deployment of CORD is now managed via higher-level tools like Kubernetes [35] and Helm [36]. Through these tools, deploying, managing and versioning the different CORD components is easier, more flexible and robust than before. This new deployment approach greatly facilitated the process of RFBising the CORD platform, mainly due to the central point of management offered by the Kubernetes and Helm tools.

Deploying CORD via the RDCL 3D tool requires the definition of several RFBs, each of which implements a specific functionality during the CORD deployment process. However, despite their differences, all RFBs can be placed in one of three categories Pre-deployment configuration RFBs, Deployment RFBs and Post-deployment configuration RFBs. Table 3 provides details these categories.

To reduce the complexity of the graph, only the Deployment RFBs are graphically illustrated by the RDCL 3D tool, while the rest are passed to the agent as metadata. Figure 13 illustrates the RFB graph that composes the CORD 6.0 platform in the RDCL 3D web interface.

As can be observed in Figure 13, the RFBs have been grouped into five major categories as children of the root RFB, depending on the role of the component they are responsible for deploying. These five categories are (1) Message Brokers; (2) Storage Mechanisms; (3) Virtual Infrastructure Managers (VIMs); (4) Orchestration and (5) Service Models. Additionally, the VIM group of RFBs is also comprised of sub-groups, namely (1) Kubernetes (2) OpenStack and (3) ONOS. While this grouping does not serve any functional role, it can significantly simplify the composition of RFB graphs. All the remaining RFBs are leaf RFBs, meaning that they are directly mapped to the execution/deployment of a micro-service. It can also be observed that all leaf RFBs are associated with a priority value, ensuring that their deployment in the execution environment is done in a preferred order. Table 4 provides a detailed list of all leaf RFBs related to the deployment of the CORD platform.

**Table 3.** RFB Categories.

| RFB Category | Details |
|---|---|
| Pre-deployment Configuration RFBs | Responsible for configuring the target infrastructure node for the subsequent deployment of the CORD platform. An example is the cloning of remote source code repositories so that CORD can be later built from. |
| Deployment RFBs | These are the core RFBs, that when executed will provide with a functional CORD deployment on the infrastructure node. The majority of them are tied to the deployment of groups of containers, through the Helm tool. An example is the deployment of the XOS orchestrator or the ONOS SDNC. |
| Post-deployment configuration RFBs | These configure a CORD deployment for a specific use case or resolve possible issues that might have risen during the deployment of the CORD platform. An example is the registration of the CORD compute node to the ONOS SDNC. |

**Table 4.** List of Deployment RFBs.

| RFB Name | RFB Role |
|---|---|
| *k8s* | Deploys *Kubernetes*, which manages all Docker containers comprising CORD. |
| *Ingress* | Deploys *Ingress*, a component related to Kubernetes which facilitates external access to services running in the cluster. |
| *rabbitMq* | Deploys *RabbitMQ*, the message broker used in OpenStack |
| *Ceph* | A set of RFBs that deploy *ceph*, the distributed storage component of OpenStack |
| *marriaDB* | Deploys *marriaDB*, the database for OpenStack |
| *memCached* | Deploys *memcached*, the distributed memory caching system for OpenStack |
| *Heat* | Deploys *Heat*, the OpenStack orchestrator. |
| *Glance* | Deploys *Glance*, the OpenStack image registry. |
| *Keystone* | Deploys *Keystone*, the OpenStack identity service. |
| *openvSwitch* | Deploys an *OpenvSwitch*, instance that connects all OpenStack VMs. |
| *Nova* | Deploys *Nova*, the computing service of OpenStack. |
| *Neutron* | Deploys *Neutron*, the networking service of OpenStack. In CORD Neutron is responsible for creating the networks connecting the different VMs. |
| *openstackChart* | Deploys the *OpenStack service synchronizer* of CORD. |
| *addressManagerChart* | Deploys a set of components, which enable public connectivity for VNFs. |
| *onosChart* | Deploys the *ONOS synchronizer* in CORD. |
| *xosChart* | Deploys the *XOS synchronizer* in CORD. |
| *ngpaasserviceChart* | Deploys the *ngpaasService synchronizer*. It also onboards the service model. |
| *localVolumeChart* | Deploys the persistent storage component of CORD. |
| *clientSetup* | Deploys the *OpenStack client* (Command Line Interface). |
| *Libvirt* | Deploys *libvirt*, the virtualisation service to be used by OpenStack |
| *Docker* | Deploys *Docker* on the target node. |
| *dockerImageDownload* | Downloads all CORD related *Docker Images* |

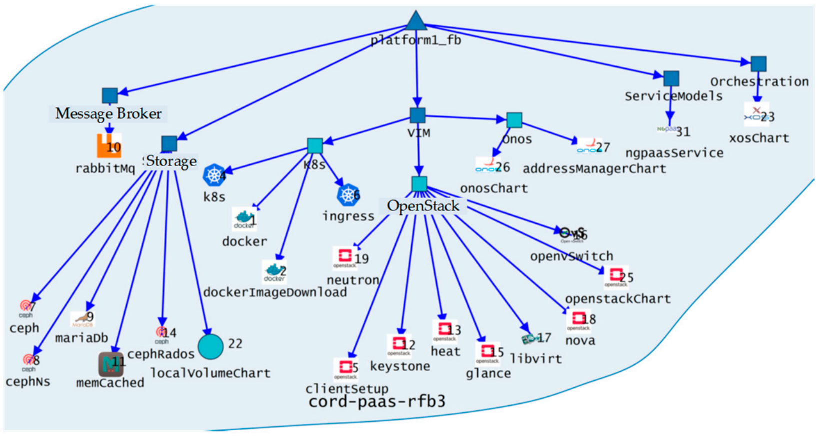


**Figure 13.** RFBisation of the CORD platform via RDCL 3D.

### 5.2. RFBisation of Services and Value-Added Services

Following a similar approach to the RFBisation of the CORD platform, all services and VAS supported by the Telco PaaS have also been partially or fully RFBized and integrated into the NGPaaS workflow. With regards to the deployment of virtual Firewalls, Routers and monitoring probes two distinct RFB types are available. (1) Instance RFB, which deploys VM based services in the OpenStack environment of CORD; (2) Network RFB, which deploys networks in the Neutron environment of CORD. Using these two RFB types, complex deployment scenarios can be expressed by the VSP. Figure 14 illustrates an RFB graph comprising of three RFB groups: (I) Router (II) Firewall and (III) Probe and 5 RFB leaves (A) Router Instance (B) Firewall Instance (C) Probe Instance (D) Data Network and (E) Monitoring Network. Following the associations between the different RFB groups and leaves, this RFB graph will provide a service chain of a Firewall and Router VNF (through a commonly shared data network), in addition this graph also expresses that out of the two VNFs the Router is to be monitored by a probe, via the monitoring network. The side-car probes of the monitoring/healing infrastructure have been RFBised, while not available at this point, it is also possible to RFBise the remaining components (e.g., ELK stack, Healer).

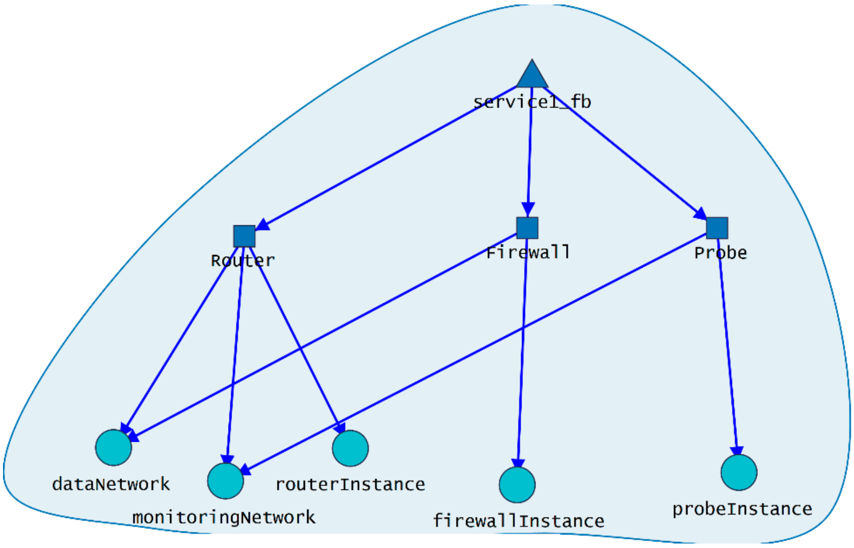
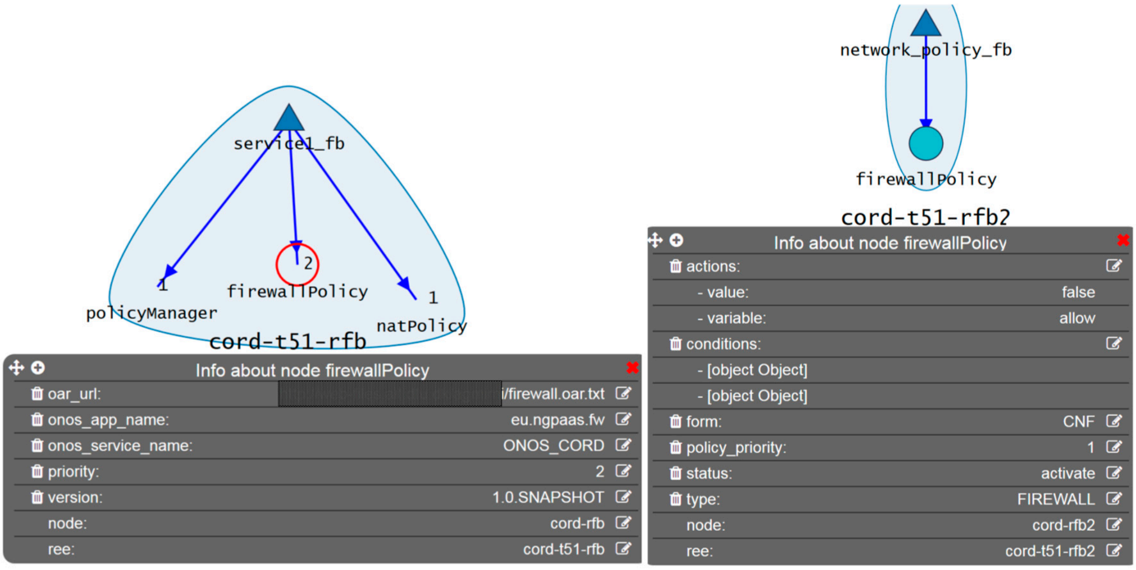


**Figure 14.** RFBisation of service deployment.

Additionally, the service provider can deploy service instances, networks and connectivity requests agnostically to the underlying technologies. It is the role of the NGPaaS OSS/BSS to translate these requests to the technology-specific APIs. More specifically for the case of the deployment of Figure 14, each of the leaf RFBs will be translated to a TOSCA recipe and then published to the TOSCA endpoint of the XOS orchestrator. Then XOS will decide how to serve this request best and translate this set of TOSCA recipes into API calls to OpenStack and ONOS. The API calls to OpenStack Nova will create the VM instances that will hold the router and firewall VNFs, while API calls to OpenStack Neutron will create the monitoring and data networks. Finally, through the dedicated REST API, XOS will instruct ONOS to populate the network with the necessary OpenFlow rules to facilitate connectivity between the VNFs.

In the Telco PaaS, the VM images used to instantiate the VNFs have all been preconfigured with an SSH channel, which provides access to their CLI interface. This way any entity with access to the deployed VNFs (e.g., VSP, end user) could interface with them and configure them on demand, without the need to re-provision them. However, if more virtual resources are required (e.g., more vCPUs), then a new VNF deployment will be necessary using the NGPaaS workflow.

The network policy framework has also been RFBised, thus allowing a VSP to either deploy policy type implementations in the policy framework or request for the enforcement of policy instances in the network. With regards to deploying the components of the policy framework only one RFB needed to be defined, namely the ONOSapp RFB. Deploying RFB graphs comprising of ONOSapp RFBs will result in the desired ONOS applications being installed and activated on the ONOS SDNC of the CORD platform. In Figure 15 (left), an RFB graph that requests the deployment of three ONOS applications is illustrated. These applications include the Policy Manager of the policy framework and two policy type implementations (Firewall and NAT). Once this RFB graph has been deployed, then the VSP can request the enforcement of Firewall and NAT policy instances. Following the example of Figure 12, the right side of Figure 15 illustrates an RFB graph that requests the enforcement of a Firewall policy instance. To facilitate the creation of RFB graphs of this type a collection of RFB types is available, one for each supported policy type (e.g., Firewall Policy Instance).



Info about node firewallPolicy

| | |
|---|---|
| oar_url: | [redacted]i/firewall.oar.txt |
| onos_app_name: | eu.ngpaas.fw |
| onos_service_name: | ONOS_CORD |
| priority: | 2 |
| version: | 1.0.SNAPSHOT |
| node: | cord-rfb |
| ree: | cord-t51-rfb |

Info about node firewallPolicy

| | |
|---|---|
| actions: | |
| - value: | false |
| - variable: | allow |
| conditions: | |
| - [object Object] | |
| - [object Object] | |
| form: | CNF |
| policy_priority: | 1 |
| status: | activate |
| type: | FIREWALL |
| node: | cord-rfb2 |
| ree: | cord-t51-rfb2 |

**Figure 15.** RFBisation of the policy framework. (**Left**) Deploys the ONOS applications comprising the policy framework (**Right**) Deploys an instance of a firewall policy.

Similarly to the deployment of the service graph of Figure 14, the deployment of network policies through the RDCL 3D tool is also technology agnostic. The service provider need only provide the necessary metadata with the deployment request (e.g., endpoints to block with the firewall policy). It is the role of the RDCL 3D tool and agent to translate this request to a JSON string and ship to the ONOS SDNC. After that, the policy framework within ONOS will validate the policy request and install the required OpenFlow rules in the network.

## 6. 5G Public Safety Use-Case

As discussed, because the NGPaaS workflow is based on the concepts of build-ship-run and build-to-order, it allows for the deployment of a wide-range of platforms, services and technologies. So far, this paper focused on the Telco PaaS use case and the VNFaaS PoC. However, to demonstrate the flexibility of the NGPaaS workflow another use case has also been considered, namely the 5G Public Safety use case. This use case is briefly demonstrated herein, through the Mission Critical Push To Talk (MCPTT) PoC.

The idea of the 5G Public Safety use case is that an end-to-end cloud native mobile network supporting the MCPTT service is provisioned on-demand, through NGPaaS. Then it can be used by firemen during an intervention at an emergency incident (e.g., a building fire). Depending on the location of the Radio Access Network (RAN), this use case could involve different scenarios. In the prototype presented herein, the fire truck is assumed to have its own, mini, data centre with the capability to host VNFs. In this way the U-Plane as well as the Cloud RAN and MCPTT related VNFs can be hosted in the fire truck, while the MCPTT control plane VNFs can be hosted in a centralised data centre. In this PoC instead of CORD, two separate Kubernetes platforms are deployed: A Core Kubernetes PaaS and an Edge Kubernetes PaaS, tailored respectively for the Core and RAN service requirements. Both Kubernetes platforms are fully deployable and configurable through the NGPaaS workflow, the same workflow used for the Telco PaaS use case. Finally, the VNFs comprising the MCPTT PoC are deployed on their corresponding platforms, similarly to that of the VNFaaS PoC. Figure 16 illustrates the architecture and functional elements of this PoC. More details are available in the technical documentation of the NGPaaS [37].

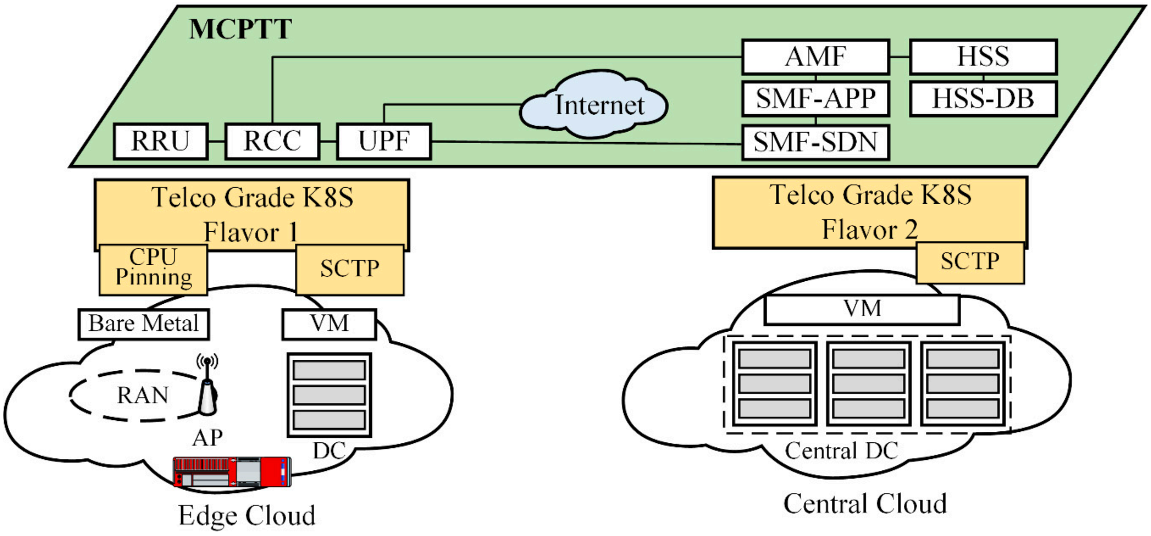


**Figure 16.** The 5G Public Safety Use Case.

## 7. Discussion and Conclusions

This paper highlights some of the inherent complexities and challenges associated with service and platform deployment in cloud-based environments, as required for the successful adoption of 5G technologies. This complexity can be mainly attributed to the broad spectrum of available infrastructure and platform technologies, each of which brings their own interfaces and communication protocols. As a possible solution to this problem, this paper introduced a novel workflow for the composition and deployment of platforms and services in multi-cloud environments. The proposed workflow follows the NGPaaS concepts of build-to-order and Build-Ship-Run, by utilising the RFB model and the RDCL 3D tool. Build-to-order implies the possibility to define custom platforms and services agnostically to the underlying technologies, based on customer demand. Build-Ship-Run implies the possibility to deploy this platform or service compositions on a wide range of execution environments, through a centralised environment (RDCL 3D tool). By following the NGPaaS workflow, platform and service deployment is greatly simplified and facilitated, since most of the underlying technologies are abstracted through generic blueprints and APIs. Finally, this paper successfully validated the proposed workflow by presenting a proof of concept scenario (Telco PaaS) that includes the composition and deployment of both a platform and a set of diverse services. The Telco PaaS use case is based on the CORD platform and represents a VNFaaS scenario, in which a VSP offers VNF services to its end customers, together with many value-added services (monitoring, healing, policy-based network management). Finally, to showcase the flexibility of the NGPaaS workflow, a second use case is also presented herein.

**Author Contributions:** Conceptualisation, P.V. and B.S.; Investigation, A.M.-K., J.S., P.V., A.B., M.M., S.V.R. and B.S.; Methodology, A.M.-K., J.S., P.V. and A.B.; Project administration, J.S. and B.S.; Resources, J.S., P.V., O.R., W.T. and B.S.; Software, A.M.-K., A.B., M.M. and S.V.R.; Supervision, J.S., P.V., O.R., W.T. and B.S.; Validation, A.M.-K., P.V., A.B., M.M. and S.V.R.; Visualisation, A.M.-K.; Writing—original draft, A.M.-K., P.V. and S.V.R.; Writing—review & editing, A.M.-K., J.S., P.V., A.B., M.M., O.R., S.V.R., W.T. and B.S.

**Funding:** This work has been performed in the framework of the NGPaaS project, funded by the European Commission under the Horizon 2020 and 5G-PPP Phase2 programs, under Grant Agreement No. 761 557 (http://ngpaas.eu).

**Conflicts of Interest:** The founding sponsors had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, and in the decision to publish the results.

## References

1. Erfanian, J.; Smith, B. Network Functions Virtualisation—White Paper on NFV Priorities for 5G. Available online: https://portal.etsi.org/NFV/NFV_White_Paper_5G.pdf (accessed on 19 May 2019).
2. Van Lingen, F.; Yannuzzi, M.; Jain, A.; Irons–Mclean, R.; Lluch, O.; Carrera, D.; Perez, J.L.; Gutierrez, A.; Montero, D.; Marti, J.; et al. The Unavoidable Convergence of NFV, 5G, and Fog: A Model-Driven Approach to Bridge Cloud and Edge. *IEEE Commun. Mag.* **2017**, *55*, 28–35. [CrossRef]
3. Van Rossem, S.; Sayadi, B.; Roullet, L.; Mimidis, A.; Paolino, M.; Veitch, P.; Berde, B.; Labrador, I.; Ramos, A.; Tavernier, W.; et al. A Vision for the Next Generation Platform-as-a-Service. In Proceedings of the 2018 IEEE 5G World Forum (5GWF), Silicon Valley, CA, USA, 9–11 July 2018; pp. 14–19.
4. Mimidis Kentis, A.; Ollora Zaballa, E.; Soler, J.; Bessem, S.; Roullet, L.; Van Rossem, S.; Pinneterre, S.; Paolino, M.; Raho, D.; Du, X.; et al. The Next Generation Platform as a Service Cloudifying Service Deployments in Telco-Operators Infrastructure. In Proceedings of the 25th International Conference on Telecommunications (ICT 2018), Saint Malo, France, 26–28 June 2018; pp. 399–404.
5. Salsano, S.; Lombardo, F.; Pisa, C.; Greto, P.; Blefari-Melazzi, N. RDCL 3D, a model agnostic web framework for the design and composition of NFV services. In Proceedings of the 2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), Berlin, Germany, 6–8 November 2017; pp. 216–222.
6. Peterson, L. "CORD: Central Office Re-Architected as a Datacenter," Open Networking Lab White Paper. November 2015. Available online: http://opencord.org/wp-content/uploads/2016/03/CORD-Whitepaper.pdf (accessed on 19 May 2019).
7. Quittek, J. "Network Functions Virtualisation (NFV); Management and Orchestration" Gs Nfv-Man 001 V1.1.1. 2014, Volume 1, pp. 1–184. Available online: https://www.etsi.org/deliver/etsi_gs/NFV-MAN/001_099/001/01.01.01_60/gs_nfv-man001v010101p.pdf (accessed on 19 May 2019).
8. Open Source MANO. Available online: https://osm.etsi.org/ (accessed on 19 May 2019).
9. 5G Tango. Available online: https://www.5gtango.eu/ (accessed on 19 May 2019).
10. Nogales, B.; Vidal, I.; Lopez, D.R.; Rodriguez, J.; Garcia-Reinoso, J.; Azcorra, A. Design and Deployment of an Open Management and Orchestration Platform for Multi-Site NFV Experimentation. *IEEE Commun. Mag.* **2019**, *57*, 20–27. [CrossRef]
11. Yilma, G.M.; Yousaf, F.Z.; Sciancalepore, V.; Costa-Perez, X. On the Challenges and KPIs for Benchmarking Open-Source NFV MANO Systems: OSM vs. ONAP. Available online: https://arxiv.org/ftp/arxiv/papers/1904/1904.10697.pdf (accessed on 19 May 2019).

12. Alvarez, F.; Breitgand, D.; Griffin, D.; Andriani, P.; Rizou, S.; Zioulis, N.; Moscatelli, F.; Serrano, J.; Keltsch, M.; Trakadas, P.; et al. An Edge-to-Cloud Virtualized Multimedia Service Platform for 5G Networks. *IEEE Trans. Broadcast.* **2019**, 1–12. [CrossRef]
13. Tarafdar, N.; Lin, T.; Ly-Ma, D.; Rozhko, D.; Leon-Garcia, A.; Chow, P. Building the Infrastructure for Deploying FPGAs in the Cloud. In *Hardware Accelerators in Data Centers*; Kachris, C., Falsafi, B., Soudris, D., Eds.; Springer International Publishing: Cham, Switzerland, 2019; pp. 9–33.
14. Chiotakis, S.; Pinneterre, S.; Paolino, M. vFPGAmanager: A Hardware-Software Framework for Optimal FPGA Resources Exploitation in Network Function Virtualization. In Proceedings of the 2019 European Conference on Networks and Communications (EuCNC), Valencia, Spain, 18–21 June 2019. Available online: http://www.virtualopensystems.com/en/research/scientific-contributions/vfpgamanager-eucnc2019/ (accessed on 19 May 2019).
15. Li, X.; Wang, X.; Liu, F.; Xu, H. DHL: Enabling Flexible Software Network Functions with FPGA Acceleration. In Proceedings of the 2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS), Vienna, Austria, 2–6 July 2018; pp. 1–11.
16. Zhang, Q.; Liu, F.; Zeng, C. Adaptive Interference-Aware VNF Placement for Service-Customized 5G Network Slices. 2019. Available online: https://pdfs.semanticscholar.org/5aea/e87f05bb4a5f08b9fae9a135863c4eaeea59.pdf?_ga=2.233200922.503865865.1558267333-871421267.1558267333 (accessed on 19 May 2019).
17. Fei, X.; Liu, F.; Xu, H.; Jin, H. Adaptive VNF Scaling and Flow Routing with Proactive Demand Prediction. In Proceedings of the IEEE INFOCOM 2018—IEEE Conference on Computer Communications, Honolulu, HI, USA, 15–19 April 2018; pp. 486–494.
18. Miao, W.; Min, G.; Wu, Y.; Huang, H.; Zhao, Z.; Wang, H.; Luo, C. Stochastic Performance Analysis of Network Function Virtualization in Future Internet. *IEEE J. Select. Areas Commun.* **2019**, *37*, 613–626. [CrossRef]
19. Wang, T.; Xu, H.; Liu, F. Multi-Resource Load Balancing for Virtual Network Functions. In Proceedings of the 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS), Atlanta, GA, USA, 5–8 June 2017; pp. 1322–1332.
20. Fei, X.; Liu, F.; Xu, H.; Jin, H. Towards Load-Balanced VNF Assignment in Geo-Distributed NFV Infrastructure. In Proceedings of the 2017 IEEE/ACM 25th International Symposium on Quality of Service (IWQoS), Vilanova i la Geltrú, Spain, 14–16 June 2017; pp. 1–10.
21. Mfula, H.; Nurminen, J.K. Self-Healing Cloud Services in Private Multi-Clouds. In Proceedings of the 2018 International Conference on High Performance Computing & Simulation (HPCS), Orleans, France, 16–20 July 2018; pp. 165–170.
22. Zhijing, L.; Zhao, B.; Zheng, H.; Ge, Z.; Mahimkar, A.; Wang, J.; Emmons, J.; Ogden, L. Predictive Analysis in Network Function Virtualization. In Proceedings of the Acm Sigcomm Internet Measurement Conference, Imc. Association for Computing Machinery, Boston, MA, USA, 31 October–2 November 2018; pp. 161–167.
23. Acker, A.; Schmidt, F.; Gulenko, A.; Kao, O. Online Density Grid Pattern Analysis to Classify Anomalies in Cloud and NFV Systems. In Proceedings of the 2018 IEEE International Conference on Cloud Computing Technology and Science (CloudCom), Nicosia, Cyprus, 10–13 December 2018; pp. 290–295.
24. Röck, C.; Kolb, S. Nucleus—Unified Deployment and Management for Platform as a Service, University of Bamberg. *Tech. Rep.* **2016**, *49*. Available online: https://www.uni-bamberg.de/pi/forschung/publikationen/16-04-nucleus-unified-deployment-and-management-for-platform-as-a-service/ (accessed on 19 May 2019).
25. Van Rossem, S.; Tavernier, W.; Colle, D.; Pickavet, M.; Demeester, P. Introducing Development Features for Virtualized Network Services. *IEEE Commun. Mag.* **2018**, *56*, 184–192. [CrossRef]
26. Veitch, P.; Broadbent, A.; Rossem, S.V.; Sayadi, B.; Natarianni, L.; Al Jammal, B.; Roullet, L.; Mimidis, A.; Ollora, E.; Soler, J.; et al. Re-Factored Operational Support Systems for the Next Generation Platform-as-a-Service (NGPaaS). In Proceedings of the 2018 IEEE 5G World Forum (5GWF), Silicon Valley, CA, USA, 9–11 July 2018; pp. 1–5.
27. Berde, B.; Van Rossem, S.; Ramos, A.; Orru, M.; Shatnawi, A. Dev-for-Operations and Multi-Sided Platform for Next Generation Platform as a Service. In Proceedings of the European Conference on Networks and Communications (EuCNC), Ljubljana, Slovenia, 18–21 June 2018; pp. 1–5.
28. ETSI NFV ISG, Network Functions Virtualisation (NFV); Network Service Templates Specification. ETSI GS NFV-IFA 014 V2.1.1 (2016–10). Available online: https://www.etsi.org/deliver/etsi_gs/NFV-IFA/001_099/014/02.01.01_60/gs_NFV-IFA014v020101p.pdf (accessed on 19 May 2019).

29. Red Hat Ansible. Available online: https://www.ansible.com (accessed on 19 May 2019).
30. Bianchi, G.; Biton, E.; Blefari-Melazzi, N.; Borges, I.; Chiaraviglio, L.; de la Cruz Ramos, P.; Eardley, P.; Fontes, F.; McGrath, M.J.; Natarianni, L.; et al. Superfluidity: A Flexible Functional Architecture for 5G Networks: G. Bianchi et al. *Trans. Emerging Tel. Tech.* **2016**, *27*, 1178–1186. [CrossRef]
31. Cisco Cloud Service Router 100V Series. Available online: https://www.cisco.com/c/en/us/products/routers/cloud-services-router-1000v-series/index.html (accessed on 19 May 2019).
32. Fortinet- FortiGate Virtual Appliance. Available online: https://www.fortinet.com/content/dam/fortinet/assets/data-sheets/FortiGate_VM.pdf (accessed on 19 May 2019).
33. Elastic Stack. Available online: https://www.elastic.co/ (accessed on 19 May 2019).
34. Ferran, C.; Mimidis-Kentis, A.; Bonourn, N.; Soler, J. Policy Framework Prototype for ONOS. In Proceedings of the 2019 IEEE Conference on Network Softwarization (NetSoft), Paris, France, 24–28 June 2019. Available online: http://orbit.dtu.dk/files/170742254/PID5809349.pdf (accessed on 19 May 2019).
35. Kubernetes. Available online: https://kubernetes.io/ (accessed on 19 May 2019).
36. The Helm Package Manager for Kubernetes. Available online: https://helm.sh/ (accessed on 19 May 2019).
37. NGPaaS Deliverable 5.6: Final platform prototype for 5G PaaS. June 2019. Available online: http://ngpaas.eu/projectoutcomes/#ngpaasdeliverables (accessed on 19 May 2019).