T-NOVA: An Open-Source MANO Stack for NFV Infrastructures

Michail-Alexandros Kourtis, Michael J. McGrath, Georgios Gardikis, Georgios Xilouris, Vincenzo Riccobene, Panagiotis Papadimitriou, Eleni Trouva, Francesco Liberati, Marco Trubian, Josep Batallé, Harilaos Koumaras, David Dietrich, *Member, IEEE*, Aurora Ramos, Jordi Ferrer Riera, José Bonnet, Antonio Pietrabissa, Alberto Ceselli, and Alessandro Petrini

***Abstract*—One of the primary challenges associated with network functions virtualization (NFV) is the automated management of the service lifecycle. In this paper, we present a full software-based management and orchestration (MANO) stack which operates with OpenStack and OpenDaylight controllers and has the in-built functionality to automate the key phases of the NFV service lifecycle, namely resource discovery and matching, service mapping, service deployment, and monitoring. The MANO stack is being implemented by the EU FP7 project T-NOVA, with the components being released as open-source software. Service mapping and service deployment solutions developed in the scope of T-NOVA are presented in detail. As a proof-of-concept, we evaluate the performance of a virtualized traffic classifier network function, demonstrating the gains of virtualized hardware acceleration.**

***Index Terms*—NFV, SDN, resource management, EPA, service mapping, monitoring, DPDK.**

I. Introduction

MANY of the latest developments in the fields of network management and service provisioning are primarily focused on Network Function Virtualization (NFV). NFV introduces a novel approach to the implementation and management of network functions. In particular, the main focus in NFV is decoupling the physical appliances from the network functions that operate on them; in other words, the transition from hardware to software -based network functions [1], [2]. This paradigm has gained significant attention from the Telecommunication Service Providers (TSPs), due to its potential to provide a significant reduction in Operating Expenses (OPEX) and Capital Expenses (CAPEX) [3], [4] as well as widening Telco's service portfolios with novel added-value software network offerings.

Software appliances in the form of Virtualized Network Functions (VNFs) run on standard high-volume (SHV) servers, which consolidate the operation and management of various network devices on a common infrastructure. VNFs provide the ability to be dynamically initiated, reconfigured and even reallocated at different locations within the network, without the requirement of installing new hardware. NFV aims at delivering a more open and extensible network environment, where the deployment of new network services becomes easier and faster.

However, in order to deliver on these expectations, various fundamental developments have to be realized, many of which are still in a state of implementation and continuous evolution. The deployment of NFV and its adoption into Telco grade systems still subsumes various open issues identified in [5]. NFV will significantly challenge current network management systems, and will require additional levels of complexity over currently deployed systems. One of the key features in this transition is the management layer, which must be capable of supporting the unique features of an NFV-enabled system. Similar works have addressed the management layer, as the Management and Orchestration system (MANO) [6]. The primary aim of MANO is to coordinate NFV Infrastructure (NFVI) resources and map them efficiently to various VNFs. In turn, VNFs can then be interconnected into chains to realize more complex Network Services (NS). A NFV NS can be seen as the evolution of a traditional telco connectivity service, as it is augmented by chains of VNFs which are dynamically inserted into network traffic paths.

The first critical challenge to be tackled by a MANO system is automated deployment. Network functions are no longer bound to a physical machine; instead they reside in

Manuscript received October 26, 2016; revised March 11, 2017 and June 30, 2017; accepted July 25, 2017. This work was undertaken under the Information Communication Technologies, EU FP7 T-NOVA project, which is partially funded by the European Commission under the grant 619520. The associate editor coordinating the review of this paper and approving it for publication was F. De Turck. *(Corresponding author: Michail-Alexandros Kourtis.)*

M.-A. Kourtis is with the National Centre for Scientific Research-Demokritos, Athens 153 10, Greece, and also with the Universidad del Pais Vasco, Lejoa 48940, Spain (e-mail: akis.kourtis@iit.demokritos.gr).

G. Xilouris, E. Trouva, and H. Koumaras are with the National Centre for Scientific Research-Demokritos, Athens 153 10, Greece.

M. J. McGrath and V. Riccobene are with Intel Labs Europe, Leixlip, Ireland.

G. Gardikis is with Space Hellas S.A., Athens 153 41, Greece.

P. Papadimitriou is with Leibniz University Hannover, Hanover 30167, Germany, and also also with the University of Macedonia, Thessaloniki 546 36, Greece.

F. Liberati and A. Pietrabissa are with the University of Rome La Sapienza, Rome 00185, Italy, also with the Space Research Group of the Consortium for the Research in Automation and Telecommunication, Roma 00185, Italy.

M. Trubian, A. Ceselli, and A. Petrini are with the Universita degli Studi di Milano, Milan 20122, Italy.

J. Batallé and J. Ferrer Riera are with FundaciÃş i2CAT, Barcelona 08034, Spain.

D. Dietrich is with Institut fÃijr Kommunikationstechnik, Hannover 30167, Germany.

A. Ramos is with Atos Spain SA, Madrid 28037, Spain.

J. Bonnet is with PT Inovacao e Sistemas SA, Aveiro 3810, Portugal.

Digital Object Identifier 10.1109/TNSM.2017.2733620

a shared resource environment. As a consequence, various considerations arise when it comes to the deployment (VNF "placement") decision, such as performance optimization, topology, resilience etc. [7]. Most of the current MANO approaches are based on the use of pre-planned configurations for the node(s) hosting the VNF workload [8]. This can limit VNF performance, as the features of the host node do not necessarily match the characteristics of the workload (e.g., data plane, control plane, heavy processing etc.). As the hosting computing infrastructures become more heterogeneous with various add-on features enabling hardware acceleration (i.e., PCIe, GPU) which can support various system enhancements and capabilities, this increases the complexity of the VNF placement decision significantly. Automated placement intelligence at deployment time increases the complexity of the MANO orchestration service. However, it results in more optimal resource allocations and increased VNF performance. A number of approaches to MANO VNF scheduling policies have been reported in [9]. Nevertheless, they generally adopt an abstracted approach to VNF placement within an NFVI in order to achieve specific goals, such as reduction of intra data center traffic [9]. These studies do not necessarily address the particular issue of mapping resources to the specific characteristics and the needs of an individual VNF workload.

In this paper, we present a tangible approach to meet this requirement, jointly addressing the challenges of resource discovery and service mapping, based on the identification of the quantity and types of resources to be allocated to a VNF at deployment time. In this respect, we generate a prototype model which expresses the allocation of resources in relation to specific levels of performance. Furthermore, specific rules are generated which can be utilized by an Orchestrator to add intelligence to a VNF deployment request in a cloud environment such as OpenStack.

In addition, a key element of service provisioning and orchestration is monitoring. Monitoring not only covers NFVI resources, but also the status of VNF services. Comprehensive monitoring also facilitates service mapping and billing/charging. It also enables fault resilience and availability, which are also critical issues, since the malfunction of a VNF will likely affect an entire network service. For all the aforementioned reasons, the development of an integrated monitoring framework for NFV, collecting metrics from physical infrastructure resources as well as specific virtualized services metrics is considered crucial for any NFV infrastructure. An effective NFV monitoring framework should expose a holistic awareness of the status and performance of the deployed services as well as the underlying infrastructure to all management entities in order to allow the latter to take proper and timely decisions. In this context, we also propose an integrated NFV monitoring solution, as a crucial component of the MANO stack.

In summary, this paper presents a novel MANO solution for NFV infrastructures, currently being developed by the EC FP7 project T-NOVA [10] and released as an open-source contribution [11]. The MANO stack is built around a novel NFV Orchestrator platform ("TeNOR") and its currently implemented modules support the management – in an automated manner – of four critical phases of the NFV service lifecycle, namely resource discovery, service mapping, service deployment and monitoring.

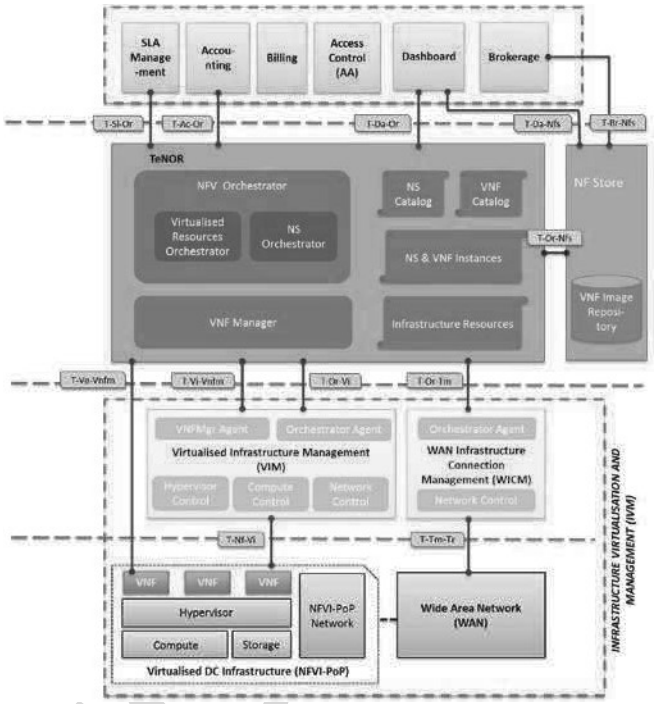


Fig. 1. T-NOVA Architecture.

The remainder of the paper is organized as follows. We initially describe the overall T-NOVA system architecture, which covers all the layers of an NFV system, also beyond MANO scope. Subsequently, we provide a detailed description and the evaluation of each component in the MANO stack. Finally, we present a proof-of-concept deployment and the evaluation of a virtualized Traffic Classifier VNF aiming to demonstrate the benefits of acceleration technologies in a NFV-enabled environment.

II. T-NOVA Overall View

A. T-NOVA System Architecture

The T-NOVA system architecture [12] inherits the majority of its concepts from the generic ETSI NFV ISG architectural model [5] and expands it with specific add-on features. The T-NOVA architecture encompasses four key architectural layers (as shown in Figure 1):

- The *NFV Infrastructure (NFVI)* layer consists of both physical and virtual nodes (high-volume servers, Virtual Machines (VMs), storage systems, switches, routers etc.) on which the network services are deployed;
- The *NFVI Infrastructure (NFVI) Management* layer includes the infrastructure management entities: the Virtualized Infrastructure Management (VIM) and the WAN Infrastructure Connection Management (WICM). T-NOVA adopts an OpenStack [18] cloud operating system for control of the compute and data-center assets and OpenDaylight [13] for the control of the network infrastructure (most of which is SDN-based);

- The *Orchestration* layer is based on the T-NOVA TeNOR Orchestrator and also includes a "Network Function Store" which is a repository for all published VNFs. The Orchestrator, along with the NFVI Management layers comprise the T-NOVA MANO stack;
- Finally, the *Marketplace* layer contains all the customer-facing interfaces and modules, which facilitate multi-role involvement and implement business-related functionalities.

T-NOVA introduces the concept of a Marketplace in an NFV framework. The aim of the Marketplace is to promote VNF service offerings and facilitating commercial activity and seamless interaction among the various business stakeholders interacting with the T-NOVA system. The T-NOVA Marketplace provides an intuitive interface to the underlying MANO stack and allows Telcos and VNF developers to publish and describe service offerings, as well as enabling customers to browse and select services, deploy and manage/monitor them. A more in-depth description of the T-NOVA Marketplace can be found in [12].

The T-NOVA Orchestrator ("TeNOR") is the core component of the T-NOVA architectural framework. Its key aim is to address NSs and VNFs lifecycle management operations over distributed and virtualized network/IT infrastructures.

Currently, there is a trend for various implementations of ETSI NFV MANO Architecture [5]. The most notable are OSM [13] and ONAP 0.

The first one is an open source project that provides a practical implementation of the reference architecture for MANO under standardization at ETSI's NFV ISG. In relation to TeNOR, both platforms provision End-to-End services but with some differences from an architecture perspective – for instance, OSM has an explicit architectural split between Resource Orchestrator and Service Orchestrator. Another aspect that sets them apart is their origin; T-NOVA implemented their components from scratch as micro-services using generic interfaces. This gives the ability to develop or replace any of TeNOR's components freely without loss of functionality, which is not possible in OSM at the moment. Moreover, TeNOR's has a specific module, Gatekeeper, which provides a common security mechanism to the interfaces between components while OSM is dependent on each component security features. Lastly, the greatest distinction between these two platforms is that TeNOR also addresses the business aspects of NFV in the VNF and NS data models, while OSM is only focused on the operationalization of NSs. With that in mind, OSM GUI and T-NOVA Marketplace also share some functionality in a graphical interface, e.g., they use a similar approach to on-board VNFs and to compose NSDs.

The latter is an offspring of the merge of the code base of Open-O [16] and ECOMP[1] [17], focusing more into bringing the gap between NFV and SDN, offering E2E service orchestration and automation. From a T-NOVA point of view, the integrated platform may be seen as an approach for a real-time, policy-driven software automation of virtual network functions that will enable software, network, IT and cloud providers and developers to rapidly create new services. In order to compare something so extensive as ONAP wants to be, we need to decompose the ONAP to it two main frameworks. One is design-time framework mostly supported by ECOMP used for the design and composition of network services, policies and VNFs. In comparison to T-NOVA this is handled by the T-NOVA Marketplace and it goes beyond the Orchestration discussion. The other framework of ONAP is Run-time execution framework, which, based on Open-O, implements all the orchestration functionalities of ONAP like TeNOR. In this context Open-O and consequently ONAP is optimized to deliver an open management platform for defining, operating and managing a wide range of products and services based upon virtualized network and infrastructure resources and software applications, whereas T-NOVA is mainly focused on VNFaaS environments and address VNFaaS-specific requirements and challenges. Hence TeNOR does not implement any type of Service Orchestration beyond the basic lifecycle operations. Additionally TeNOR supports only VIMs based on Openstack, however it could easily be extended to support other VIM technologies too.

Other approaches include the OpenBaton [15] which is a result of a synergy between Fraunhofer Fokus and TUB for implementing and ETSI compliant MANO solution. Compared to TeNOR, both platforms share the common features of: Automated Scaling, Multi-PoP deployment of a virtual Network Service, a VNF and Service store catalog, and a functional dashboard. However, in T-NOVA the orchestrator dashboard is more feature rich, and a rich marketplace including accounting, billing as well as brokerage are included. Additionally, T-NOVA orchestrator supports Transport Network management between POPs and supports advanced service function chaining capabilities (through NetFloc). Open Baton acquiesce SDN management to external modules not influenced by the orchestrator itself. Open Baton's external module Network-Slicing-Engine performs network slicing and ensuring QoS. TeNOR has a more extensive Service life-cycle management support built into the framework, whereas till the point of writing, Open Baton has extensive support for VNF life-cycle management only.

A thorough comparison with the above orchestration solutions was not possible due to the different maturity level of each solution. However, distinguishable features of TeNOR include microservices architecture, the multi-pop/multi-administration domain support and the support for modular service mapping and placement algorithms.

TeNOR interacts with the Marketplace, which is the external-endpoint responsible for establishing the business and operational management of T-NOVA. Besides the Marketplace, TeNOR also interfaces with the VIM for managing the data center network/IT infrastructure resources, as well as with the WAN Infrastructure Connection Management (WICM) for the WAN elements connectivity management. Finally, TeNOR interacts with the VNF itself to provide appropriate lifecycle management.

[1] AT&T Project.

To support NFV lifecycle management operations, the following categories are defined, most of which are inherited from the ETSI NFV ISG recommendations:

- NS Catalogue: represents the repository of all the on-boarded NSs (already deployed NSs) in order to support NS lifecycle management;
- NS Descriptor (NSD): contains the service description, including Service Level Agreements, deployment flavors, references to the virtual links (VLs) and the constituent VNFs, Virtual Network Function Forwarding Graph (VNFFG);
- Virtual Link Descriptor (VLD): contains the description of the virtual network links that compose the service (interconnecting the VNFs);
- VNF Forwarding Graph Descriptor (VNFFG): contains the NS constituent VNFs, as well as their deployment in terms of network connectivity;
- VNF Catalogue: represents the repository of all the on-boarded VNFs in order to support their lifecycle management;
- VNF Descriptor (VNFD): contains the VNF description, including its internal decomposition based on Virtual Network Function Components (VNFCs), deployment flavors and references to the virtual links (VLDs);
- Software images of the VMs located in the NFVI layer;
- NS and VNF Instances: represents the repository of all the instantiated NSs and VNFs, which can be updated/released during the lifecycle management operations;
- Infrastructure Resources: represents the repository of the available/reserved/allocated NFVI resources as abstracted by the VIM across operator's infrastructure domains. Furthermore, it also includes the resources available/reserved/allocated in the WAN segment.

B. TeNOR – the T-NOVA NFV Orchestrator

Network Service lifecycle provisioning is managed by the T-NOVA orchestration platform, TeNOR, whose NFV Orchestrator functionality module is split into two main core submodules: Network Service Orchestrator (NSO) and Virtualized Resource Orchestrator (VRO) as shown in Figure 2.

The NSO responsibility is for the management of the NS lifecycle and associated procedures, including:

NSes and VNFs on-boarding: management of Network Services deployment templates, also known as NS Descriptors and VNF Packages, as well as of the NSs instances topology (e.g., create, update, query, delete VNF Forwarding Graphs). On-boarding of a NS includes registration in the NS catalogue, therefore ensuring that all the templates (NSDs) are stored.

NS instantiation: trigger instantiation of NS and VNF instances, according to triggers and actions captured in the on-boarded NS and VNF deployment templates. In addition, management of VNF instantiation, in coordination with Virtualized Network Function Managers (VNFMs) as well as validation of NFVI resource requests from VNFMs, as they may impact NSs, e.g., scaling process.

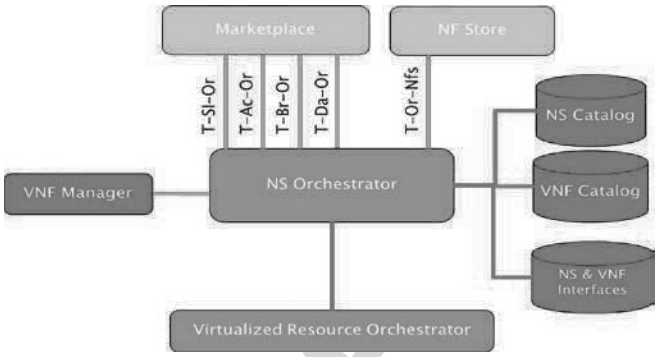


Fig. 2. TeNOR NS Orchestrator (Internal & External) Interactions.

NS update: supports NS configuration changes of varying complexity such as changing inter-VNF connectivity or the constituent VNFs;

NS supervision: monitoring and measurement of the NS performance and correlation of the acquired metrics for each service instance. Data is obtained from the IVM layer (performance metrics related with the virtual network links interconnecting the network functions) and from the VNFM (aggregated performance metrics related with the VNF;

NS scaling: increases or decreases the capacity of a NS according to per-instance and per-service auto-scaling policies. NS scaling can imply either increase/decrease of a specific VNF capacity, creation/termination of new/old VNF instances and/or increase/decrease of the number of connectivity links between the network functions;

NS termination: release of a specific NS instance by removing the associated VNFs and associated connectivity links, as well as the virtualized infrastructure resources.

The VRO is responsible for the management of the underlying physical computing and network resources. In this context, the VRO interfaces with: (i) the Virtualized Infrastructure Manager(s) that are managing the resources in each NFVI's Points-of-Presence (NFVI-PoP) where the VNFs are deployed; (ii) the WAN Infrastructure Managers (WIM) that manage the WAN resource allocation. The VRO also tracks all of the underlying infrastructure via the infrastructure repository. In the case of a NS deployment request, the VRO is responsible for allocating the resources required according to the requirements imposed by each NS.

From an external perspective, the key NSO interactions with the Marketplace for operational and business management purposes are as follows:

- Exchange of provisioning information (e.g., requests, modifications/ updates, acknowledgements) with respect to NSes (through the T-Da-Or interface);
- Providing the Orchestrator with information on each NS instance SLA agreement. In turn the Orchestrator sends SLA-related metrics to the Marketplace (through the T-Sl-Or interface);
- Delivery of usage accounting information to the Marketplace with respect to VNFs and NSs (through the T-Ac-Or interface);
- Providing the Orchestrator with information regarding NS composition. The Orchestrator sends the Marketplace

information on the available VNFs (through the T-Br-Or interface).

Internally, the NSO has the following communication points:

- NS Catalogue: collects information about the NSs (NSD), including the set of constituent VNFs, interconnecting network links (VLD) and network topology information (VNFFGD);
- VNF Catalogue: stores the Virtual Network Function Descriptor document during the on-boarding procedures;
- NS and VNF Instances: stores information about the NS instances status;
- VRO: exchanges management actions in relation to virtualized resources and/or connections, either within the data center scope (e.g., compute, storage and network) and/or on the transport network segment;
- VNFM: exchanges lifecycle management actions related with the VNFs.

The next sections discuss in detail the specific components of the T-NOVA MANO stack which enable the management of the critical phases of an NFV service lifecycle.

III. Resource Discovery, Selection, Matching

IT Cloud computing environments aim to maximize the utilization (efficiency) of compute resources and employ highly automated operations to schedule and manage workloads. To achieve these goals significant flexibility is required, which is realized through the abstraction of the underlying infrastructure environment. By hiding the heterogeneity of the underlying compute, storage and network resources, workloads can be placed or migrated more easily. As a result, application performance is generally delivered on a best effort basis.

In contrast, Telco cloud applications have a unique set of characteristics, such as requirements for low latency, high packet throughput, low jitter, network setup, and teardown constraints. In addition, these workloads may have key performance indicators (KPIs) defined in SLAs which must be met within an operational context. The underlying infrastructure hosting the network workloads plays an important role in helping to achieve these KPIs. This is particularly true in the context of NFV, where, the goal is to achieve near line rate performance in a virtualized infrastructure environment.

These requirements result in a number of key challenges which must be addressed in a Telco cloud environment. First, the resources available within the cloud must be discoverable and their associated descriptions stored in an accessible repository. Second, it is necessary to understand and map the characteristics of the workload (e.g., data plane workloads versus control plane workloads) and their affinities to resources types (e.g., Open vSwitch (OVS), SR-IOV, DPDK capable network interface card) and the quantity of allocated resources (e.g., number of vCPU, amount of RAM, etc.). This is necessary to prevent the over allocation of resources which have no tangible benefit on workload performance. When allocating resources where more than one option is available (e.g., type of network connection for a VM), it is important to allocate resources which deliver a quantifiable benefit to workload performance. This allows Telco service providers to better monetize their infrastructure by ensuring that workloads are deployed on platforms where they will benefit from the specific resource available and to improve workload consolidation. Finally, the platform configuration is also an important consideration. For example, when deploying a workload with high packet throughput characteristics using PCIe aware Non-Uniform Memory Access (NUMA) pinning can have a significant impact on VNF performance. NUMA is a multiprocessing memory design which provides separate memory for each processor, enabling the processor to bind to the memory resource and gain faster access to it.

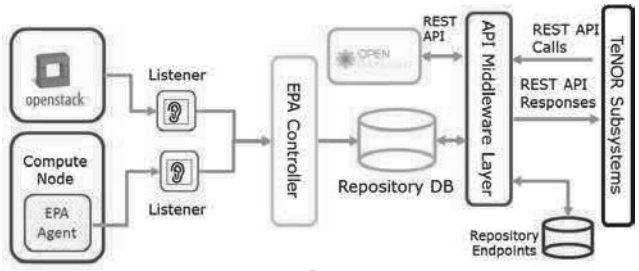


Fig. 3. T-NOVA resource repository sub-system.

A. NFVI Resource Discovery and Awareness

In order for the VNFs to achieve a performance which is close or similar to a counterpart dedicated hardware implementation, appropriate exploitation of platform features, in terms of both hardware and software, within the NFVI environment is critical. T-NOVA accomplished this through the development of an infrastructure repository subsystem which is part of the TeNOR developed by the project. The design of the repository subsystem addresses the challenges of assimilating infrastructure related information from sources within the NFVI / VIM layers, namely the cloud infrastructure and data center network environments within the T-NOVA system. This subsystem comprises a number of key elements including a data model, resource information repositories and access mechanisms to the information repositories. The subsystem also augments the information provided by cloud and SDN environments through a resource discovery mechanism.

Enhanced Platform Awareness (EPA) enables fine-grained matching of resources available in the NFVI to VNFs in order to better optimize the network services being provided. To enable EPA, the available platform features must be identified and exposed. As shown in Figure 3 a subsystem comprising of EPA agents running on the compute nodes in the NFVI collect and report the required platform information. When platform updates are available from the EPA agents, notification messages are sent to a controller via a specific listener service. Upon receipt of messages from the EPA listener service, data files sent by the EPA agents to a storage directory are processed by the controller and used to update the central repository database. A listener service is also used to intercept and pipeline infrastructure related messages in OpenStack

and to update the repository database via the controller. The infrastructure repository database is implemented as a graph database. The database provides a hierarchical relationship in the form of semantically relevant connections between the nodes stored as a link. The use of a graph conveniently maps to the hierarchical structure of the compute, storage and network elements within the NFVI. An Open Cloud Compute Interface (OCCI) [20] compliant middleware layer provides a common interface to the resource information stored in the graph databases. Additionally, the middleware API provides an abstracted single access point to physical network information available from OpenDaylight through its REST API's. The middleware layer also features a graph database which is used to store the endpoint information of OpenStack services for each NFV-PoP under the control of TeNOR.

B. Characterization of VNF Affinities With Infrastructure Resources

For any VNF type workload it is necessary to build a picture of the VNF's affinities for compute resource allocations and platform specific features. It is also important to contextualize the Telco performance aspects as they are generally multi-faceted in nature. Key influencing factors such as packet sizes and network connection type were identified together with key measures for determining performance, such as packet throughput, etc. Various deployment configurations were investigated including key network technologies (virtualized packet switching and packet acceleration), storage and compute (core pinning, NUMA pinning, heterogeneous compute resources BIOS configurations etc.).

For example, the effect of noisy neighbors was investigated on a virtualized traffic classifier deployment, when exploiting different types of resources. A noisy neighbor is a term commonly used in cloud computing environments to describe a co-tenant workload that can negatively impact the performance of other workloads through heavy utilization of specific resources such as network bandwidth, CPU, memory etc. As a result, the noisy neighbor effect causes other workloads such as VNFs that share the cloud infrastructure to suffer from unpredictable performance. To investigate the impact of different configurations on the performance of the virtualized traffic classifier VNF both test cases followed the standard benchmarking methodology proposed by RFC 2544 [21] to measure network throughput. The test cases used seven different packet sizes (64, 128, 256, 512, 1024, 1280 and 1518 bytes) with trial duration of 60 seconds for the throughput measurements. In the first test case, the deployment did not include noisy neighbors, whereas in the second one the throughput was measured in the presence of noisy neighbors who added additional overhead to the CPU of the node hosting the VNF. Figure 4 shows the throughput results obtained using a packet size of 1280 bytes. In this case a throughput rate of 3.5Gbps corresponds to $\sim$ 2.7 million frames per second.

In the first test case, the results show that huge pages (memory, e.g., 1GB) [22] have no measurable impact on the performance, whereas core pinning improved performance, especially if used in conjunction with the "isolcpus"

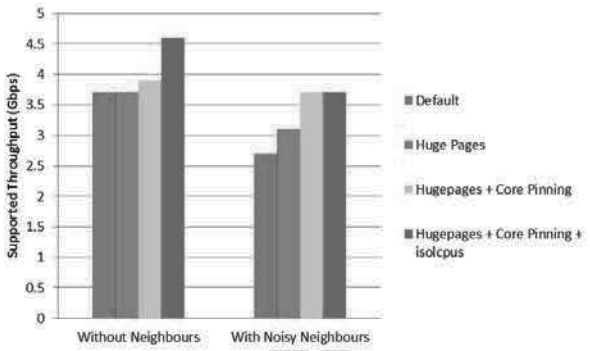


Fig. 4. Packet throughput performance of a VNF with different compute configurations.

Grub option, by up to 24%. The isolcpus option provides CPU isolation from the general kernel symmetric multi-processing (SMP) balancing and scheduler algorithms. This has the effect of isolating the cores from user-space tasks. In the second test case (with noisy neighbors) the base line performance with the default configuration is 2.7Gbps, which is lower than the first test case scenario. The reason for the reduction in throughput is the additional computation overhead placed on the CPU servicing requests from the additional VM's. In this test case the impact of the huge pages on performance is clearly visible (up to 14% improvement in throughput). With the use of core pinning it is possible to further improve the performance up to an additional 23% increase in throughput.

C. Discovery of Optimal Resource Allocations for VNFs

As outlined earlier once the infrastructure resources are discoverable and have been characterized, the final element is workload characterization. The goal of the characterization process is to match the appropriate resource types and quantities in order to achieve the required KPIs for a given deployment. The current approach to deploying workloads such as VNFs in cloud computing environments is based on a static predefined allocation of resources. These allocation are typically defined in an abstract manner, i.e., the resource request is generically defined simply as a quantity of vCPUs, memory, storage and network connections without reference to specific characteristic or attributes of these resource types. These characteristics could include support for specific instruction sets such as AES-NI by a CPU or intelligent offload support in a network card etc. This results in the over allocation of resources and secondly does not match the types of resources available effectively for each type of workload. To address this issue, a VNF workload characterization framework was developed which can be used to interrogate all potential configuration permutations for a defined set of configuration variables and associated value ranges. The framework collects the metrics data for each permutation and processes the data using a machine learning environment to generate a set of deployment rules. The generalized process for workload characterization and deployment rule generation is shown in Figure 5. To support the process a framework was

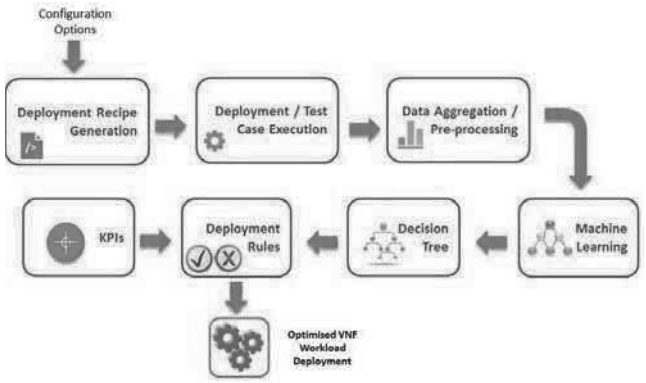


Fig. 5. VNF deployment rule generation process.

```
port:
 type: OS::Neutron::Port
 properties:
  network: { get_param: network }
  binding:vnic_type: #vnic_type
# allowed values: [normal, direct]
  fixed_ips:
      - subnet: { get_param: subnet }

server_flavor:
 type: OS::Nova::Flavor
 properties:
  disk: #disk    # allowed values: [1024, 2048, ...]

  ram: #ram     # allowed values: [1024, 2048, ...]
  vcpus: #vcpus     # allowed values: [1, 2, 3, ...]
  extra_specs: {
      "hw:cpu_policy": "#core_pinning_enabled",
          # allowed values: [shared, dedicated]
      "hw:cpu_threads_policy": "#core_pinning_mode",
          # allowed values: [isolate, prefer]
      "hw:mem_page_size": "#hugepages"
          # allowed values: [small, large]
                }
```

Fig. 6. Code extract from a generic heat template to be manipulated before submission to Heat.

developed which takes as its input a configuration that defines the resources and quantity ranges of interest, e.g., from 1 to 10 vCPUs, from 1 to 8GB of RAM, etc. The configuration file then translates the configuration options into a set of deployment templates for the target virtualization environment, e.g., OpenStack. The workload under test is automatically deployed using the deployment templates. Test cases are automatically executed for each configuration, e.g., measurement of network throughput (RFC2544) and corresponding telemetry data are collected and stored. The test cases focus on the quantification of VNF performance indicators. When all deployments templates have been deployed the data collected is then aggregated and used by a machine learning algorithm (e.g., C4.5 [23]) to generate a model which relates performance indicators (e.g., throughput) for a VNF with respect to different allocations of resources. These relationships can then be expressed in the form of deployment rules (e.g., throughput = 5Gbps can be achieved with an allocation of vCPU = 4, RAM = 8GB, network connection = SR-IOV, etc.). Additional details on the methodology and application can be found in [24].

The set of rules automatically generated by the framework can be exploited to generate deployment recipes, (e.g., Heat Orchestration Templates (HOT) [25] for OpenStack where parameters are derived according to the target KPIs for the workload or service being deployed. An example of HOT syntax to implement automatic generation of Heat templates is shown in Figure 6. A Heat template with generic parameters is created which are replaced at run time with the values calculated by applying the rules generated through the framework, according to the workloads target KPIs.

Collectively, resource discovery, workload and technology characterization, automated deployment rule and recipe generation for VNF deployment provide a unique capability within the T-NOVA system.

IV. Service Mapping

The term Service Mapping (SM) refers to the problem of dynamically mapping virtualized infrastructure resources to NSes, respecting the constraints posed by: (i) the current availability of network infrastructure resources, (ii) the type and amount of resources demanded by the services to be mapped and (iii) SLA specific needs. This problem has been attracting increasing attention over the last number of years. Early research approaches described in the literature, such as [26]–[28], have presented cloud platform implementations that allow NSes to be arbitrarily integrated into VMs, without considering the functionalities of a service chain. Other research efforts [29], [30] provide heuristics for mapping chains of NS inside Data Center (DC) networks, with the aim of maximizing KPIs such as inter-rack traffic. Several approaches in the literature are devoted to the theme of service mapping modelling. Among them, [31] developed a graph-based technique, called Tenant Application Graph (TAG), to accurately capture bandwidth requirements for VMs being deployed, avoiding overprovisioning inefficiencies associated with previous methodologies. The same work proposes a min-cut and knapsack algorithm for SM, whose driving rationale is to maximize co-location of VMs linked by edges with stringent link bandwidth requirements.

The deployment of NSes over multiple DCs, i.e., mapping NS chains over inter-DC networks finally enables the wide-area deployment of NSs. This type of problem is often compared to the virtual network embedding problem [32], which has inspired several service mapping formulations. However, the rich variety of proposed embedding algorithms cannot be directly applied to service chains due to the different NS types, policies exercised by the middlebox operators, and the changing traffic rates caused by some NSs. Among the advanced service mapping algorithms, [33] proposes NS node and virtual link mapping based on cost metric optimization (aimed at node/link load balancing) and shortest path computation. Guerzoni *et al.* [34] propose an Integer Linear Programming (ILP) algorithm based on undirected graph modelling of the infrastructure and services. Interestingly, the authors implement and discuss the so called "lookahead" property, i.e., the simultaneous mapping of bunches of NSes. Similarly, in [35] a mixed ILP strategy is presented, which is aimed at minimizing path latencies, number of used resources and maximizing the remaining node/link resource availability. Also soft-computing techniques have been proposed, such as the approach outlined in [36]. Other SM approaches similar to the ones presented above can be found in [8], [32], and [37].

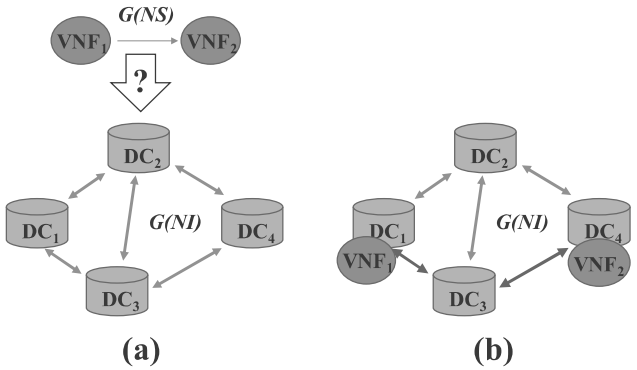


Fig. 7. Illustrative example of SM problem (a) and its solution (b).

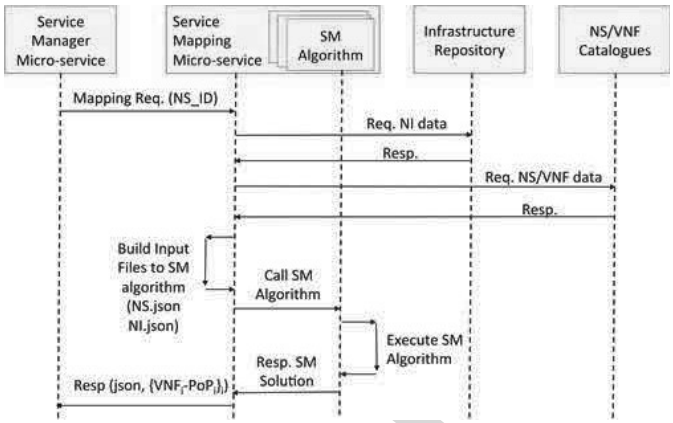


Fig. 8. Reference SM sequence diagram.

The service mapping problem addressed in T-NOVA focuses on the assignment, based on optimization techniques of the VNFs composing each NS request to the interconnected DCs composing the Network Infrastructure (NI). The requests for NSs arrive to the system dynamically: they are not known in advance and the SM algorithm has to identify an optimal assignment for each request, or discard it, on demand. As for any online optimization problem, the objective function which is optimized when solving each mapping request has to implicitly model the true overall target function, i.e., maximize the number of accepted requests.

In this paper, we present an ILP-based approach for the SM problem, and particularly for the assignment of VNFs to NFV-PoPs. In this respect, the objective of the proposed ILP is the maximization of accepted NS requests. The particular SM method has been implemented and integrated into the T-NOVA orchestrator. Our SM method is different from the ILP-based methods proposed in [38] and [39]. Specifically, in our system the service mapping module is entirely executed by the infrastructure provider and the optimization objective is tailored to its policy (i.e., maximization of request acceptance and hence revenue). In [38] and [39], the assignment of VNFs to PoPs is carried out by a broker which primarily aims at minimizing the client's expenditure. T-NOVA investigated three SM approaches, two based on ILP and one based on reinforcement learning. This paper will focus on the ILP approach that is currently fully integrated in the T-NOVA system. This choice depends mainly on the fact that a solution based on a ILP model is more flexible than one based on a tailored heuristic, especially if T-NOVA will be further elaborated in future projects. Our aims were threefold: (a) to have a model able to consider as much as possible resources and kinds of resource constraints, (b) to take into account more than one objective function and (c) to give a scalable solution even when non commercial solvers were used.

A. Problem Modeling

The SM problem addressed in T-NOVA is outlined in Figure 7(a) which shows a NS composed by two VNFs, a NI composed by four interconnected DCs and their corresponding graphs. Figure 7(b) shows a possible solution of the corresponding SM problem. VNF_1 has been assigned to DC_1, VNF_2 has been assigned to DC_4 and the arc connecting VNF_1 and VNF_2 has been assigned to the blue path from DC_1 to DC_4, through DC_3.

As shown in Figure 7, each NS is modeled as a directed graph $G(NS) = (V, A)$ in which each vertex in the set V, say h, represents a VNF, and each arc in A, say (h, k), represents a link connecting two VNFs. Similarly, the NI is modeled as a directed graph $G(NI) = (V^I, A^I)$ in which each vertex, say p, in the set V^I represents a PoP, and each arc in A^I, say (p, q), represents the network connection between two PoPs. To each NS, a set P of paths is associated connecting all the pairs of VNFs (each path $\pi \in P$ being a sequence of arcs in the graph $G(NS)$). Each path in P is associated with a maximum allowed delay Δ_π, to account for service latency requirements. On the other hand, an actual delay value δ_{pq} is associated to each arc (p, q) in $G(NI)$. Finally, NT (respectively, LT) denotes the set of all available node (link) resource types. In this regard, RR_h^t denotes the resource requirement of VNF h with respect to resource type t, and RA_p^t the amount of resources of type t available at PoP p. Link resource requirements RR_{hk}^t and availabilities RA_{uv}^t are defined similarly. At the beginning of each mapping event, $G(NI)$, NT, LT, RA_p^t and RA_{uv}^t are updated through the NI API Middleware Layer, while $G(NS)$, P, Δ_π, RR_h^t and RR_{hk}^t are populated according to the information included in the NSD and VNFD of the service to be mapped.

Based on the modelling above, the ILP-based mapping approach implemented in the T-NOVA system is outlined in the following sections. Before that, the next section briefly explains the reference architecture and sequence diagrams governing on the SM module functions in according to the T-NOVA architecture.

B. Reference Architecture and SM Sequence Diagram

Figure 8 shows the basic sequence diagram governing SM operations. A SM microservice for hosting the SM algorithms has been developed based on RESTful Web services. The microservice primarily interfaces with the Infrastructure Repository and the VNF/NS Service Catalogues, which are the database sources for populating, at each mapping time, the graph-based SM models detailed in the previous paragraphs.

Each time a NS must be mapped, a request is made to the SM micro-service, which, in turn, queries the infrastructure repository and the service catalogue for the input information needed to build and solve the SM problem.

C. ILP-Based Network Service Mapping

In this section, we discuss the ILP-based service mapping approach proposed and developed in T-NOVA. To facilitate network service mapping, we decompose the problem into two levels: (i) the assignment of VNFs to DCs (referred to in the following as *first level SM*) and (ii) the placement of VNFs onto servers in the selected DC, as well as the assignment of the NF-graph edges onto physical paths within the NI. This approach significantly reduces the complexity of the service mapping problem.

In the following, we present the ILP formulation for the first level service SM problem. For the second level, T-NOVA relies directly on OpenStack primitives. We use binary variables y_p^h to express the assignment of VNF h to the DC p, whereas the binary variables x_{pq}^{hk} indicate whether the link (h, k) in graph $G(NS) = (V, A)$ has been mapped onto a path among DCs in graph $G(NI) = (V^I, A^I)$ which uses the link (p, q).

The ILP for first level mapping is formulated as follows:

Problem (*ILP based Service mapping*)

Minimize

$$\alpha \sum_{h\in V}\sum_{p\in V^I} c_p^h y_p^h + \beta \sum_{\pi\in P}\sum_{(h,k)\in\pi}\sum_{(p,q)\in A^I} \delta_{pq} x_{pq}^{hk} + \gamma \sum_{t\in LT}\sum_{(h,k)\in A}\sum_{(p,q)\in A^I} RR_{hk}^t x_{pq}^{hk} \quad (1)$$

Subject to

$$\sum_{p\in V^I} y_p^h = 1 \qquad \forall h \in V \quad (2)$$

$$\sum_{q\in V^I} x_{pq}^{hk} - \sum_{q\in V^I} x_{qp}^{hk} = y_p^h - y_p^k \quad \forall (h,k) \in A,\ \forall p \in V^I \quad (3)$$

$$\sum_{(h,k)\in\pi}\sum_{(p,q)\in A^I} \delta_{pq} x_{pq}^{hk} \le \Delta_\pi \qquad \forall \pi \in P \quad (4)$$

$$\sum_{(h,k)\in A} RR_{hk}^t x_{pq}^{hk} \le RA_{pq}^t \ \forall (p,q) \in A^I, \quad \forall t \in LT \quad (5)$$

$$\sum_{h\in V} RR_h^t y_p^h \le RA_p^t \ \forall p \in N^I, \quad \forall t \in NT \quad (6)$$

$$y_p^h \in \{0,1\} \quad \forall h \in V, \quad \forall p \in V^I \quad (7)$$

$$x_{pq}^{hk} \in \{0,1\} \quad \forall (h,k) \in A,\ \forall (p,q) \in A^I \quad (8)$$

The objective function (1) aims at implicitly modelling the actual service mapping target that is the maximization of the number of accepted NS requests. It is a weighted sum of three components: (i) the cost of assigning VNFs to DCs, (ii) the overall delay and (iii) the overall resource link usage, as derived by assigning the links among VNFs to path among DCs. In particular, given a service request and a network infrastructure, for each VNF h composing the service, and for each DC p composing the network infrastructure, we introduce a virtual cost (mapping a VNF to a DC) c_p^h which is used to weight the *cost* of assigning h to p in terms of maximization of accepted requests. The weighting parameters , and have been tuned according to the results of an experimental campaign.

Constraints (2) ensure that each VNF h is mapped exactly to one DC. Constraints (3) ensure that for a given pair of VNFs h and k assigned to DCs p and q, respectively, there is a path in the network infrastructure graph $G(NI)$ connecting p to q to which the edge (h, k) has been mapped. Constraints (4) impose the satisfiability of a SLA based on the delay thresholds for each path in the P set. Constraints (5) impose the link resource limit of the inter DC connections for each link resource type t in the resource type set LT. Constraints (6) impose the node resource limit of the DC for each node resource type t in the resource type set NT. Constraints (7) and (8) enforce binary domain for the variables.

Our ILP formulation contains a polynomial number of variables and constraints, and is therefore suitable for optimization by general-purpose solvers. Hence, we performed a computational evaluation of the method. For this task a synthetic simulator was built, which works as follows. First, the simulator is populated with the network infrastructures and the NSes included in the dataset [40] described in **Error! Reference source not found.**, a popular reference in the literature of mapping algorithms. The dataset consists of 210 base instances, partitioned in 7 classes of increasing size networks, each one including 30 instances. Any instance contains a graph describing the NI and a number of smaller graphs describing the NSes. NS graphs belong to a limited number of topologies, representing different service types. NI (respectively, NS) graph nodes are annotated with data representing available (respectively, required) nodes and links resources. Furthermore, for each NS node a vector of compatibilities to NI nodes is given, indicating which mappings are feasible due to specific resources needed by the NS. We considered each arc in the NS graph (and only them) as critical paths, whose delay constraints have to be respected.

We model the arrival of NS requests as a Poisson[2] process, whose average inter-arrival time is denoted as λ. We also assume the duration of NS allocation requests to be random, following a normal distribution whose mean is denoted as μ and whose standard deviation is denoted as σ. A simulation run consists of selecting a particular base instance (that is, a NI and the corresponding pool of possible NSs), and then (a) iteratively and randomly selecting one NS, an inter-arrival time and a duration for the corresponding allocation request, (b) asking the mapping algorithm to find a suitable allocation of the drawn NS to the NI, (c) eventually updating NI resources either according to the mapping provided by the algorithm or because of expired services. The simulation ends as soon as the sum of inter-arrival times exceeds a given time horizon τ, that represents the time length of the simulation.

[2]We model the arrival of NS requests as a Poisson process, based on the assumption that requests come from independent sources (i.e., when these independent requests are aggregated, they form a Poisson process). Modeling the arrival rate of virtual network and NS requests is an established assumption in [31], [32], and [54].

1) Tuning of Model Parameters: In a preliminary round of tests, we tried to assess the behavior of the service mapping algorithm as the parameters of the corresponding model change. We considered only instances whose NI span 20 NFV-PoPs,[3] whereas the number of NS was set to 40. We considered eight configurations, one for each possible choice of either value 1.0 or value 0.0 for each of the model parameters α, β and γ. We considered for a given base instance all NS allocation requests to arrive following the order in which they appear in the instance file, with a negligible fixed interarrival time, and a duration equal to τ. That is, all NS allocation requests arrive one after another, they are either rejected or allocated, and in the second case the assigned resources are never released. We always set the simulation length $\tau = 168$ hours (that is, one week), the average allocation request duration $\mu = 24$ hours, and the corresponding standard deviation $\sigma = 2$ hours. The first 20 allocation requests were always accepted in all configurations. In the following 20 ones we registered that moving from $\beta = 0.0$ to $\beta = 1.0$ substantially improves the acceptance rate; the same applies moving from $\gamma = 0.0$ to $\gamma = 1.0$. Moving from $\alpha = 0.0$ to $\alpha = 1.0$ has still some impact, when combined with settings $\beta = 1.0$ and $\gamma = 1.0$. This matches our modelling aim, in which objective terms related to β and γ directly affect allocation feasibility, while that related to α is useful only for diversification, and therefore inter PoP balancing. Overall, setting all parameters to 1.0 generated the best results.

2) Evaluating Solvers Scalability as the NI Size Increases: First, we verified that the approach is computationally effective enough to be embedded in TeNOR's service mapping module. We considered using either the open source solver GNU GLPK or IBM's commercial CPLEX solver. A time limit of 60s was given to each solver run. The simulations were ran with instances with up to 100 NI nodes.

We analyzed two scenarios: mild and high average NI load, setting in the former case $\lambda = \mu$ / (0.50 * (L / l)), and in the latter case $\lambda = \mu$ / (0.75 * (L / l)), where L is the overall available CPU resources in the NI, and l is the average amount of CPU resource required by each NS in the corresponding pool. That is, if CPU's were the only scarce resource, and NS node fragmentation was possible, in the mild (respectively, high) average NI load scenario we would expect to have about 50% (respectively, 75%) of overall CPU resources always allocated. We considered two performance measures: the percentage of accepted NS allocation requests, and the average computing time per allocation request.

We first observe that computing time is not a critical issue: the commercial solver CPLEX never exceeds the time limit, and the average response times is as low as 0.3s even for large NIs. The open-source solver GLPK yields computing times that are one order of magnitude larger than those of CPLEX; this was expected, giving the benchmarking results from the literature. Still, the average query time is always less than 3s. Allocation rejection is almost always produced as the result of the solver detecting infeasibility (timeout is observed on average in 0.3% of the runs, and only for GLPK); rejections are on average faster to report than allocations. In summary, both solvers scale well in terms of computing time: embedding either with CPLEX or GLPK would likely yield systems whose performance bottleneck is not the service mapping algorithm.

At the same time, GLPK and CPLEX provide almost identical results, even if GLPK incurs more often (0.3% of the runs) in timeouts. To further check the efficiency of the solvers when very fast response is required, we repeated the tests by setting a time limit of 3s instead of 60s in the computing time. The reduction of the time limit did not yield any notable inefficiency. Therefore, efficient mappings can be generated on relatively short timescales.

Also in terms of acceptance rate, embedding either CPLEX or GLPK, even by imposing tight time limits, has no significant impact on the overall performance of the system.

3) Solvers Scalability as Datacenters Load Increases: Second, we analyzed the performances of our ILP based approach as the average DC load changes. In particular, we restricted our tests to instances whose NIs containing 20 nodes. Simulations were considered where the average CPU load (ratio of the requested CPU resource over the available one) of the NI nodes, denoted as δ, ranges from 0.1 to 1.2, considering each step of 0.1 points, and setting $\lambda = \mu/(\delta * (L \ / \ l))$ for each node.

The acceptance rate drops almost linearly from 100% to 50% as the average load increases, with robust system stability even under high levels of stress. Average load seems to have very little effect on the solvers' computing time requirements. The reduction of the time limit does not yield any notable inefficiency in terms of mapping between the two solvers.

V. Service Deployment

Provisioning and instantiation represent the major lifecycle management activities within the T-NOVA system, both for network services and the VNFs composing them. Once the Service Mapping, as described in Section IV, has been completed, the TeNOR system has visibility to the PoPs where the service and functions must be provisioned. Apart from instantiation, lifecycle management also includes termination of VNFs. Figure 9 shows the sequence diagram for NS Provisioning from the TeNOR perspective, including the high-level interactions of the workflow.

From a general perspective, NS Provisioning is composed of the following phases (refer to Figures 1 and 2 for the T-NOVA architecture components):

- If required, TeNOR creates a connectivity resource in the WICM, by sending the required information, such as the identifier, the Network Access Point, or a descriptor. It receives back the corresponding VLAN identifiers for both the ingress and egress network points created.
- TeNOR starts the provisioning process, which is realized by means of a HEAT [25] orchestration template which is sent to the VIM. The VIM is responsible for instantiating the different components described in the template.

[3]We consider that an NFV infrastructure spanning 20 PoPs provides sufficient scale for most NSes, as it can meet NS geolocation requirements (i.e., PoPs will be deployed at different locations) and offer large computing capacity for the deployment of VNFs.

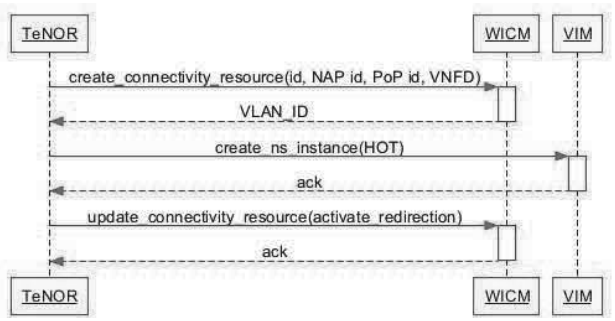


Fig. 9. High-level TeNOR NS Provisioning Sequence Diagram.

- TeNOR updates the connectivity resource (i.e., WICM) created to adapt the traffic redirections to the instances created (both VMs and virtual networks) by the previous call.

This process enables complete automation of the end-to-end service provisioning including one or more NFVI-PoPs and the transport network managed by the WICM.

A. NS Creation Process

The creation of an NS instance, i.e., create_ns_instance (HOT) as shown in Figure 9, requires specific actions for various micro-services within TeNOR. First, it is worth noting that provisioning one NS comprehends the deployment of all the VNFs composing it, and as a consequence it may become a complex process to automate. In essence, the activity inside step 2 of the previous workflow can be summarized as follows: (i) whenever a new NS instantiation request reaches TeNOR from the Marketplace, the NS Provisioning micro-service retrieves the information from the NS Descriptor; (ii) with that information the Service Mapping returns an ordered list of feasible PoPs where to allocate the VNFs composing the service; (iii) with the PoP and inter-PoP connectivity information, a loop is started and each VNF that is part of the NS is then forwarded to the corresponding VNF Manager, who is responsible for the deployment of the VNF at the VIM layer (i.e., creation of the HEAT template and communication with the corresponding VIM service); and (iv) finally, upon each successful NS instantiation request, an SLA is created and the necessary parameters are subscribed for monitoring (refer to Section VII).

B. Performance Evaluation of NS Provisioning

In terms of evaluating the performance of NS Provisioning within T-NOVA the most important metrics to be assessed is the overall NS Provisioning time. This is the total time from receipt of the request, the time required to complete processing, the Service Mapping time and VNF instantiation in the NFVI. Thus, the first analyzed time is the mapping algorithm, which calculates the best available location in the NFVI. Then, the VNF instantiation time, which includes parsing information from the VNFD, generation of the HEAT template, and sending the request to the OpenStack Heat service to instantiation the VMs in the NFVI.

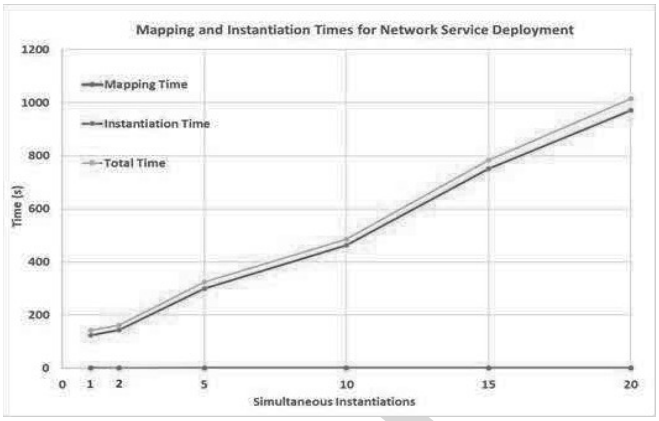


Fig. 10. Mapping and instantiation time for multiple network services instantiation.

The graph shown in Figure 10 depicts the time required for multiple (1, 2, 5, 10, 15 and 20 requests simultaneously) instantiations of a network service (consisting of only one Virtual Machine that requires 2vCPUs, 2GB RAM and 20GB of disk space). The mapping time is the total time required for the mapping algorithm to define the resource allocation. The instantiation time starts when the NS provisioning knows where to allocate the VNFs until the instance is available in OpenStack. This time consists of the tenant and user creation, the creation of the Heat template, and downloading the VNF image from Glance.

As we can see, the mapping time varies insignificantly in all simultaneous instantiations because the REST call/response to the Mapping module requires less than 100ms. However, the instantiation time increases as the number of simultaneous instantiations increments through time, as a result of the orchestrator sending a unique request to OpenStack to deploy network resources and the VNF image for each instance. In the next step, OpenStack deploys each instance simultaneously resulting in a demanding request for OpenStack to allocate the resources simultaneously. The OpenStack allocation operations become the bottleneck, and as a result given 5 simultaneous instantiation requests, the final time required for deployment is doubled. Additionally, regarding the statistical significance of the aforementioned graph the standard deviation values were calculated for the corresponding times series. The results were: 0.01544 for the Mapping time, 341.4665 for the Instantiation time, and 351.334 for the Total time.

VI. Infrastructure and Service Monitoring

In an NFV environment, proper infrastructure and service monitoring is crucial for a number of reasons: i) to maintain an integrated picture of NFVI status and resources, ii) to enable proper service mapping, iii) to facilitate accounting/billing and SLA management and iv) to trigger (proactive or reactive) actions in fault scenarios.

IT and network monitoring has been an area of active research area for more than three decades. As a result, a wide variety of tools and concepts are available which can be

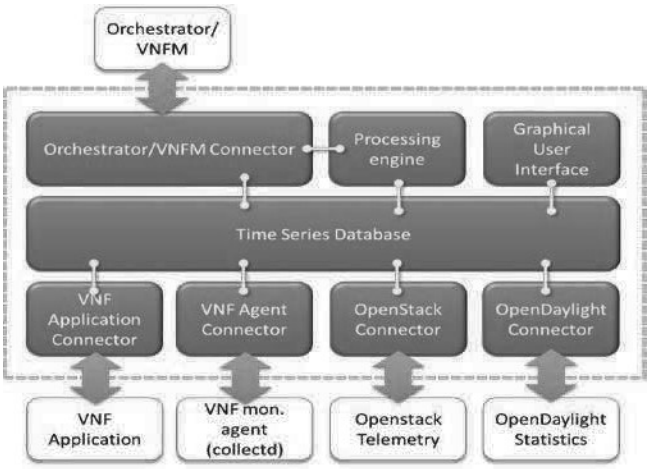


Fig. 11. Functional architecture of the T-NOVA VIM Monitoring Manager.

exploited and/or adapted for use in an NFV environment. However, not all of them can be used as-is. A monitoring framework especially tailored for an NFV environment should fulfil specific requirements, such as:

- Interfacing with VIM control modules (e.g., OpenStack, OpenDaylight etc.)
- Interfacing with VNFs and/or the VNFM to retrieve VNF-specific (application) metrics
- Scalability. The scalability requirement is crucial, since, in a large-scale NFV environment, the MANO stack could be overloaded with millions of measurements per second, coming from all parts of the NFV infrastructure and services. The monitoring framework needs to be able to filter out unnecessary data and to identify specific events (and, if needed, alarms) to be communicated to high-level orchestrating entities.

In order to fulfil these requirements, in T-NOVA, we followed the approach of designing and developing a dedicated lightweight monitoring framework tailored to the needs of an operational NFV environment. For its design and implementation, we studied and exploited some of the features and concepts of existing state-of-the-art monitoring and support frameworks for virtualized environments, such as Monasca [42] and Gnocchi [43]. We also examined emerging OPNFV projects on NFV monitoring, such as Doctor [44] and Prediction [45].

Our approach is based on a dedicated monitoring entity within the VIM layer, the VIM Monitoring Manager (VIM MM), whose functional architecture is shown in Figure 11.

The VIM MM collects, aggregates and processes data from various sources, namely:

- The physical and virtual infrastructure, including compute nodes, storage servers, as well as network switches and routers. For this purpose, the VIM MM directly interfaces with the infrastructure controllers (OpenStack and OpenDaylight), polling their monitoring APIs (Ceilometer/Telemetry API [46] and Statistics respectively).
- The VMs on which the VNFs are running. The aim here is to augment the monitoring capabilities of the system, compared with the limited set of metrics which Ceilometer natively provides. For this purpose, we introduce a monitoring agent in each VM. The agent is based on the popular collected-core module [47] which can be directly installed with minimal overhead. The use of the collected agent provides access to a much wider set of metrics from the guest OS compared to OpenStack's Ceilometer data collection service, such as detailed memory and network usage metrics, load information and process statistics. It also collects metrics at a significantly higher resolution, improving the response time of the monitoring system. The agent can be either preinstalled in the VNF image or installed upon deployment via the Heat template; both options are supported. However, in the case where inclusion of a monitoring agent in the VNF is not desirable (e.g., not allowed/supported by the VNF developer), the system can also work in agent-less mode, solely relying on data from Ceilometer.
- The VNF applications themselves. VNFs are expected to provide application-specific metrics, such as the number of sessions, number of flows, response latency etc. In order to facilitate the exposure of these metrics to the monitoring system, we created a lightweight SDK which can be used by VNF developers to communicate metrics directly to the VIM MM (in addition to the information exchanged with the VNFM).

All collected metric are stored in a time-series database, which is more appropriate for time-series data than a standard relational database (DB). We selected InfluxDB [48] for this purpose. By aggregating all data into a performant DB and relying on periodical feeds, we can simplify workflows, reduce inter-component signaling and thus eliminate the need for a message queue, which is commonly used in monitoring frameworks.

A back-end processing engine processes the data in real-time in order to generate events and alarms. Events/alarms are generated either via pre-defined thresholds (set by the Orchestrator) or by dynamically detecting deviations from "normal" VNF operations. This is achieved using anomaly detection algorithms. The anomaly detection functionality remains in development.

All information is exposed to the Orchestrator via a REST-based API, which supports the following operations:

- Standard GET request for on-demand retrieval of a specific metric or groups of metrics.
- Subscribe operation for periodic "pushing" of specific metrics.
- Subscription to alarms using pre-defined thresholds.

In all cases, the values communicated or used as alarm thresholds may refer to either instantaneous samples, or statistical aggregates (min, max, average, median etc.) apart from the programmatic interface (API), the metrics are also visualized in an intuitive, Web-based graphical user interface (GUI), which is based on Grafana [49]. A screenshot of the GUI, is shown in Figure 12. In this case, information from a virtual

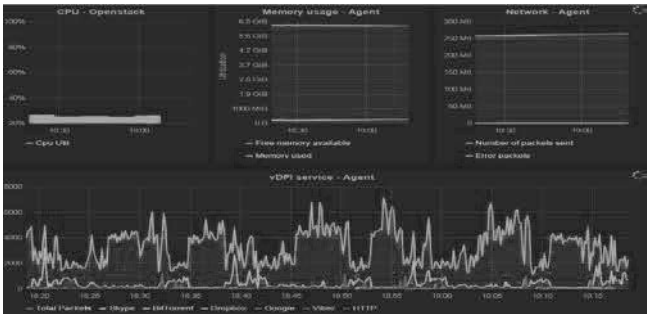


Fig. 12. GUI screenshot aggregating metrics from various sources to monitor a vTC VNF.

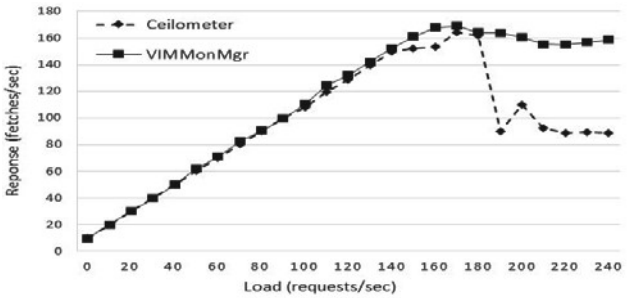


Fig. 13. VIM Monitoring Manager performance.

Traffic Classifier (vTC) VNF (see next section) is displayed. The following metrics are integrated into a single view:

- VNF CPU utilization, retrieved from OpenStack Ceilometer
- VNF memory usage and network traffic (cumulative packet count), as reported by the guest OS via the collected agent.
- VNF-specific metrics (packet rate of different applications detected by the vTC, e.g., Skype, Bittorrent, Dropbox, Google, Viber etc.), as reported by the vTC service.

We also evaluated the scalability and behavior of the VIM MM's northbound API. To emulate frequent data exchange with the Orchestrator, we used a number of concurrent GET requests, specifying a single metric (CPU load) from the vTC VNF. We used the httperf software [50] to generate synthetic HTTP GET requests at various rates and measured the rate of responses received. Then, we repeated the procedure, this time directly polling Ceilometer for the same metric.

The two sets of measurements were carried on platforms with similar hardware capabilities. The results are depicted in Figure 13.

From the results obtained it is clear that the VIM MM can expose metrics with a performance level which is comparable to native Ceilometer. It also appears to exhibit better stability when in overload situations (at more than 160 requests/sec for the given hardware configuration). It should be noted that this relatively low saturation point is due to the restricted hardware resources of the platform; in any case, the results are comparative to Ceilometer and are not absolute.

All components of the T-NOVA VIM monitoring framework, along with the appropriate documentation, have been released as open-source [51] under a GPL license, as part of the T-NOVA MANO stack. The released version has been integrated in Docker containers, in order to facilitate deployment in heterogeneous environments.

VII. Use Case: Deploying a Hardware-Accelerated Virtual Traffic Classifier (vTC)

A. vTC Architecture and Acceleration Techniques

In this last section, we consider as an indicative use case of the T-NOVA MANO stack the deployment of an NFV service comprising a virtual Traffic Classifier (vTC) which takes advantage of specific hardware acceleration features commonly available in an NFVI. The vTC was also developed in the context of the T-NOVA project and is being currently used as part of the OPNFV Yardstick [52] project which is focused on NFV workload characterization.

The vTC is designed to analyze in real-time network traffic, to recognize specific applications and to prioritize each traffic flow according to application priority policies. As this operation generates significant workload, it is highly recommended to employ specific hardware acceleration features for efficient traffic processing. In the following paragraphs we briefly describe the techniques which were employed.

The Linux network stack, on which the vTC has been developed is commonly used as a basis for cloud networking solutions. Its primary goal is the provisioning of a general purpose network stack for a fully functional operating system rather than a stack that is specifically designed for high packet throughput performance. Therefore, a standard Linux network stack cannot scale to the performance level often required for a software network appliance.

As I/O performance is critical in cloud infrastructures, virtualization optimizations are required in order to maximize the utilization of computer system resources. Single Root I/O Virtualization (SR-IOV) [53] is a specification released by the PCI-SIG, which defines hardware enhancements that reduce hypervisor's interactions with a VM, in order to improve its data processing performance. An SR-IOV enabled device is capable of spawning various "light" instances of PCI functions, named Virtual Functions (VFs). Each VF can be pinned to a VM, granting direct access to its physical resources. Exploiting this feature, multiple VMs can share the same physical PCI device resources, thus achieving higher performance with no significant additional CPU overhead. As SR-IOV provides direct access and control to the system's hardware resources, it enables the efficient allocation of low-level network resources.

The vTC VNF is used as a use-case in T-NOVA to demonstrate the performance enhancements provided by the SR-IOV specification and the packet processing acceleration framework DPDK. The vTC comprises of two Virtual Network Function Components (VNFCs), namely the Traffic Inspection engine and Classification and Forwarding function. The two VNFCs are implemented in respective VMs. The proposed Traffic Classification solution is based upon a Deep Packet Inspection (DPI) approach, which is used to analyze a small

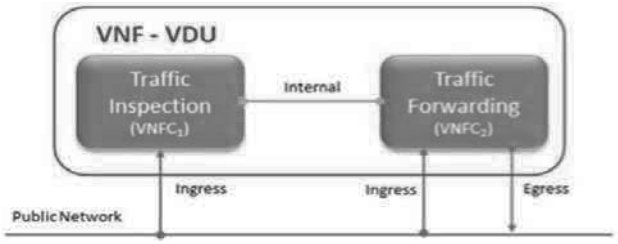


Fig. 14. Virtual Traffic Classifier VNF component architecture.

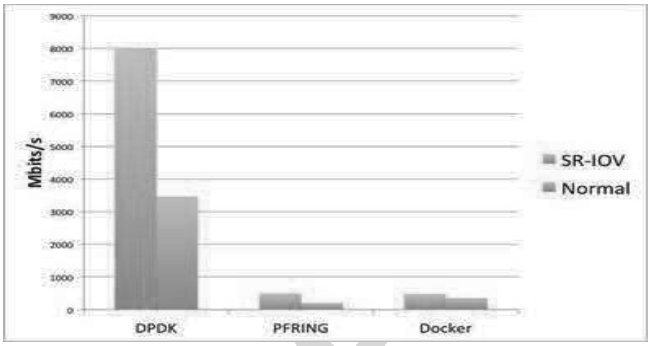


Fig. 15. Comparative results for the vTC VNF in various setup environments.

number of initial packets from a flow in order to identify the flow type. After the flow identification step no further packets are inspected. The Traffic Classifier follows the Packet Based per Flow State (PBFS) in order to track the respective flows. This method uses a table to track each session based on the 5-tuples (source address, destination address, source port, destination port, and the transport protocol) that is maintained for each flow. The architectural overview of the vTC VNF is shown in Figure 14.

The vTC utilized various technologies in order to offer a stable and highly performant VNF. The vTC implementation is based upon the open source nDPI library [54]. The packet capturing mechanism is implemented using a number of technologies in order to investigate the respective trade-offs between performance and modularity. The packet handling/forwarding technologies evaluated were:

PF_RING [55]: PF_RING is a set of library drivers and kernel modules, which enable high-throughput, packet capture and sampling. For the vTC the PF_RING kernel module library was used, which polls the packets through the LINUX NAPI. The packets are copied from the kernel to the PF_RING buffer and then analyzed using the nDPI library.

Docker [56]: Docker is a form of Linux container which provides a self-contained execution environment, that provides isolated CPU, memory, block I/O, and network resources based on sharing the kernel of a host operating system. In order to investigate the pros and cons of the container technology, the vTC was developed also as an independent container application. The forwarding and the inspecting of the traffic are also performed using PF_RING and nDPI, and they are modified accordingly to function properly in a containerized environment.

DPDK: In order to fully exploit the system's resources, both in the network and computational domains, and at the same time enhance and facilitate the implementation of intensive network applications, Intel has developed the Data Plane Development Kit (Intel®DPDK) [57]. DPDK comprises of a set of libraries that support efficient implementations of network functions through access to the system's network interface card (NIC). DPDK offers network function developers a set of tools to build high speed data plane applications. DPDK operates in polling mode for packet processing, instead of the default interrupt mode. The polling mode operation adopts the busy-wait technique, continuously checking for state changes in the network interface. This mitigates interruption in packet processing, as it bypasses the kernel, efficiently consuming CPU cycles, which leads to increased packet throughput [58]. Using DPDK network packet ingress and egress is faster in comparison to the standard Linux kernel network stack as applications are supported in userspace, thus bypassing kernel network stack bottlenecks. A DPDK-enabled version of the vTC has been implemented in order to optimize the packet-handling and processing for the inspected and forwarded traffic, by bypassing the kernel space. The analyzing and forwarding functions are performed entirely in user-space which enhances the vTC performance.

B. Validation and Assessment Using the T-NOVA MANO Components

As a validation test, we used the T-NOVA MANO components described in the previous sections to discover resources for the vTC, assigned the appropriate resources, deployed the service and monitored it under load conditions. During each iteration, the hardware acceleration capabilities of the vTC were properly detected (via the corresponding entries in the VNF Descriptor (VNFD) document) and correctly matched to the compute nodes which had these specific capabilities.

A series of comparison tests between the DPDK, PF_RING, and Docker versions of the vTC in physical and virtualized environments were carried out. Traffic for the tests was generated using the open source PktGen traffic generator [59] which can generate up to 10Gbps of network traffic. The results collected illustrated the behavior of each implementation based on linear scaling network traffic load up to 10Gbps at line rate. Traffic statistics were collected from the VNF at one second intervals and were post processed for performance evaluation.

The results in Figure 15 show a clear improvement in the vTCs' performance when DPDK is utilized. The gap in performance between the physical and the virtualized solution shows further optimization is required in order for to achieve line rate. Additionally, it is clear that despite the use of SR-IOV the network kernel stack remains the bottleneck in the packet processing path.

As shown in Figure 14 the SR-IOV/DPDK version achieves approximately 81% of the physical DPDK testbed packet transmission performance. The PF_RING kernel version displayed saturation effects at 500Mbps with an 87.5% throughput reduction in comparison to the DPDK version. The results also indicate that DPDK's performance in the virtualized

scenario is degraded, approximately 19% in comparison to the corresponding tests performed on non-virtualized hardware. This performance degradation is a result of the additional hypervisor overhead in the virtualized environment with respect to the packet processing overhead.

VIII. Conclusion

We presented the components of an operational NFV MANO stack being developed by the T-NOVA project and released as an open-source project [11]. The current version of the T-NOVA MANO system facilitates the automation of the key NFV service lifecycle steps, specifically resource discovery and matching, service mapping, service deployment and monitoring. Of particular significance are the capabilities within the system to discover, map and exploit hardware platform-specific features to optimize performance, as demonstrated in the vTC assessment activities. The complete MANO stack has been designed in a fully modular manner under a service-oriented architecture, so that each of the components described in this paper can also be individually exploited on an individual basis, if necessary.

Additional functional capabilities to be implemented include service auto-scaling (scaling in/out), SLA monitoring, fully automated traffic steering/service function chaining (SFC) and anomaly detection.

References

[1] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee, "Network function virtualization: Challenges and opportunities for innovations," *IEEE Commun. Mag.*, vol. 53, no. 2, pp. 90–97, Feb. 2015.

[2] R. Guerzoni, "Network functions virtualisation: An introduction, benefits, enablers, challenges and call for action, introductory white paper," in *Proc. SDN OpenFlow World Congr.*, Jun. 2012, pp. 1–16.

[3] J. Wu, Z. Zhang, Y. Hong, and Y. Wen, "Cloud radio access network (C-RAN): A primer," *IEEE Netw.*, vol. 29, no. 1, pp. 35–41, Jan./Feb. 2015.

[4] *C-RAN: The Road Towards Green RAN. White Paper. Version 2.5*, China Mobile Res. Inst., Beijing, China, Oct. 2011.

[5] *ETSI GS NFV 002 V1.2.1: Network Functions Virtualisation (NFV); Architectural Framework*, ETSI Ind. Specification Group (ISG) NFV, Sophia Antipolis, France, Dec. 2014. [Online]. Available: http://www.etsi.org/deliver/etsigs/NFV/001099/002/01.02.0160/gsNFV002v010201p.pdf

[6] *ETSI GS NFV 003 V1.2.1: Network Functions Virtualisation (NFV); Terminology for Main Concepts in NFV*, ETSI Ind. Specification Group (ISG) NFV, Sophia Antipolis, France, Dec. 2014. [Online]. Available: http://www.etsi.org/deliver/etsigs/NFV/001099/003/01.02.0160/gsNFV003v010201p.pdf

[7] S. Oechsner and A. Ripke, "Flexible support of VNF placement functions in OpenStack," in *Proc. 1st IEEE Conf. Netw. Softwarization (NetSoft)*, London, U.K., 2015, pp. 1–6.

[8] K. Giotis, Y. Kryftis, and V. Maglaris, "Policy-based orchestration of NFV services in software-defined networks," in *Proc. 1st IEEE Conf. Netw. Softwarization (NetSoft)*, London, U.K., 2015, pp. 1–5.

[9] M. Yoshida, W. Shen, T. Kawabata, K. Minato, and W. Imajuku, "MORSA: A multi-objective resource scheduling algorithm for NFV infrastructure," in *Proc. 16th Asia–Pac. Netw. Oper. Manag. Symp. (APNOMS)*, 2014, pp. 1–6.

[10] Accessed on Aug. 4, 2017. [Online]. Available: http://www.t-nova.eu/

[11] Accessed on Aug. 4, 2017. [Online]. Available: https://github.com/T-NOVA

[12] G. Xilouris *et al.*, "T-NOVA: A marketplace for virtualized network functions," in *Proc. Eur. Conf. Netw. Commun. (EuCNC)*, Bologna, Italy, Jun. 2014, pp. 1–5.

[13] Accessed on Aug. 4, 2017. [Online]. Available: https://osm.etsi.org/wikipub/index.php/Main_Page

[14] *Open Network Automation Platform (ONAP)*. Accessed on Aug. 4, 2017. [Online]. Available: https://www.onap.org

[15] Accessed on Aug. 4, 2017. [Online]. Available: http://openbaton.github.io/

[16] Accessed on Aug. 4, 2017. [Online]. Available: https://www.open-o.org/

[17] Accessed on Aug. 4, 2017. [Online]. Available: http://about.att.com/content/dam/snrdocs/ecomp.pdf

[18] Accessed on Aug. 4, 2017. [Online]. Available: https://www.openstack.org/

[19] Accessed on Aug. 4, 2017. [Online]. Available: https://www.opendaylight.org/

[20] Accessed on Aug. 4, 2017. [Online]. Available: http://occi-wg.org/

[21] S. Bradner and J. McQuaid, "Benchmarking methodology for network interconnect devices," Internet Eng. Task Force, Fremont, CA, USA, RFC 2544, Mar. 1999.

[22] Accessed on Aug. 4, 2017. [Online]. Available: https://www.kernel.org/doc/Documentation/vm/hugetlbpage.txt

[23] J. R. Quinlan, *C4.5: Programs for Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann, 1993.

[24] V. Riccobene, M. J. McGrath, M.-A. Kourtis, G. Xilouris, and H. Koumaras, "Automated generation of VNF deployment rules using infrastructure affinity characterization," in *Proc. 2nd IEEE Conf. Netw. Softwarization (NetSoft)*, Seoul, South Korea, Jun. 2016, pp. 226–233.

[25] OpenStack. (2015). *Heat Orchestration Template (HOT) Guide*. [Online]. Available: http://docs.openstack.org/developer/heat/template_guide/hot_guide.html

[26] H. Ballani, P. Costa, T. Karagiannis, and A. Rowstron, "Towards predictable datacenter networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 4, pp. 242–253, 2011.

[27] J. Lee *et al.*, "CloudMirror: Application-aware bandwidth reservations in the cloud," in *Proc. 5th USENIX Workshop Hot Topics Cloud Comput.*, San Jose, CA, USA, 2013, pp. 1–6.

[28] C. Guo *et al.*, "SecondNet: A data center network virtualization architecture with bandwidth guarantees," in *Proc. ACM Conf. Emerg. Netw. Exp. Technol. (CoNEXT)*, Philadelphia, PA, USA, 2010, pp. 1–12.

[29] A. Gember, R. Grandl, A. Anand, T. Benson, and A. Akella, "Stratos: Virtual middleboxes as first-class entities," Dept. Comput. Sci., Univ. Wisconsin-Madison, Madison, WI, USA, Tech. Rep. 1771, 2012.

[30] S. Oechsner and A. Ripke, "Flexible support of VNF placement functions in OpenStack," in *Proc. 1st IEEE Conf. Netw. Softwarization (NetSoft)*, London, U.K., 2015, pp. 1–6.

[31] J. Lee *et al.*, "CloudMirror: Application-aware bandwidth reservations in the cloud," in *Proc. 5th USENIX Workshop Hot Topics Cloud Comput.*, San Jose, CA, USA, 2013, pp. 1–6.

[32] A. Fischer, J. F. Botero, M. T. Beck, H. De Meer, and X. Hesselbach, "Virtual network embedding: A survey," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 4, pp. 1888–1906, 4th Quart., 2013.

[33] R. Riggio, T. Rasheed, and R. Narayanan, "Virtual network functions orchestration in enterprise WLANs," in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manag. (IM)*, Ottawa, ON, Canada, 2015, pp. 1220–1225.

[34] R. Guerzoni *et al.*, "A novel approach to virtual networks embedding for SDN management and orchestration," in *Proc. Netw. Oper. Manag. Symp. (NOMS)*, Kraków, Poland, 2014, pp. 1–7.

[35] S. Mehraghdam, M. Keller, and H. Karl, "Specifying and placing chains of virtual network functions," in *Proc. IEEE 3rd Int. Conf. Cloud Netw. (CloudNet)*, 2014, pp. 7–13.

[36] V. Abedifar, M. Eshghi, S. Mirjalili, and S. M. Mirjalili, "An optimized virtual network mapping using PSO in cloud computing," in *Proc. 21st Iran. Conf. Elect. Eng. (ICEE)*, 2013, pp. 1–6.

[37] R. Mijumbi *et al.*, "Network function virtualization: State-of-the-art and research challenges," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 236–262, 1st Quart., 2016.

[38] D. Dietrich, A. Abujoda, and P. Papadimitriou, "Network service embedding across multiple providers with nestor," in *Proc. IFIP Netw.*, Toulouse, France, May 2015, pp. 1–9.

[39] D. Dietrich, A. Rizk, and P. Papadimitriou, "Multi-provider virtual network embedding with limited information disclosure," *IEEE Trans. Netw. Service Manag.*, vol. 12, no. 2, pp. 188–201, Jun. 2015.

[40] Algorithms and Complexity Group. (2015). *Virtual Network Mapping Problem Instances*. Accessed on Dec. 15, 2015. [Online]. Available: https://www.ac.tuwien.ac.at/research/problem-instances/

[41] J. Inführ and G. R. Raidl, *Introducing the Virtual Network Mapping Problem With Delay, Routing and Location Constraints*. Heidelberg, Germany: Springer, 2011, pp. 105–117.

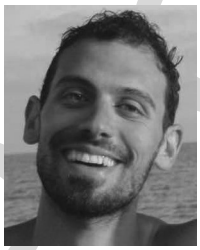
[42] *Monasca Monitoring-As-a Service for OpenStack.* Accessed on Aug. 4, 2017. [Online]. Available: https://wiki.openstack.org/wiki/Monasca
[43] *Gnocchi: TDBaaS for OpenStack.* Accessed on Aug. 4, 2017. [Online]. Available: https:// wiki.openstack.org/wiki/Gnocchi
[44] *OPNFV Doctor Project (Fault Management).* Accessed on Aug. 4, 2017. [Online]. Available: https://wiki.opnfv.org/doctor
[45] *OPNFV Prediction Project (Data Collection for Failure Prediction).* Accessed on Aug. 4, 2017. [Online]. Available: https://wiki.opnfv.org/prediction
[46] *OpenStack Telemetry API v2 (CURRENT).* Accessed on Aug. 4, 2017. [Online]. Available: http://developer.OpenStack.org/api-ref-telemetry-v2.html
[47] *Collectd, the System Statistics Collection Daemon.* Accessed on Aug. 4, 2017. [Online]. Available: https://collectd.org/
[48] *InfluxDB: An Open-Source, Distributed, Time Series Database With No External Dependencies.* Accessed on Aug. 4, 2017. [Online]. Available: http://influxdb.com/
[49] *Grafana: An Open Source, Feature Rich Metrics Dashboard and Graph Editor for Graphite, InfluxDB & OpenTSDB.* Accessed on Aug. 4, 2017. [Online]. Available: http://grafana.org/
[50] Accessed on Aug. 4, 2017. [Online]. Available: https://github.com/httperf/httperf
[51] Accessed on Aug. 4, 2017. [Online]. Available: https://github.com/T-NOVA/vim-monitoring
[52] Accessed on Aug. 4, 2017. [Online]. Available: https://wiki.opnfv.org/yardstick
[53] SR-IOV. *PCI Special Interest Group.* Accessed on Aug. 4, 2017. [Online]. Available: http:// www.pcisig.com/home
[54] L. Deri, M. Martinelli, T. Bujlow, and A. Cardigliano, "nDPI: Open-source high-speed deep packet inspection," in *Proc. Int. Wireless Commun. Mobile Comput. Conf. (IWCMC)*, Nicosia, Cyprus, Aug. 2014, pp. 617–622.
[55] A. Cardigliano, L. Deri, J. Gasparakis, and F. Fusco, "vPF_RING: Towards wire-speed network monitoring using virtual machines," in *Proc. IMC*, Nov. 2011, pp. 533–548.
[56] Docker. Accessed on Aug. 4, 2017. [Online]. Available: https://www.docker.com/
[57] DPDK Organization. (2015). *DPDK: Data Plane Development Kit.* Accessed on Aug. 4, 2017. [Online]. Available: http://dpdk.org/
[58] K. Salah and A. Qahtan, "Implementation and experimental performance evaluation of a hybrid interrupt-handling scheme," *Comput. Commun.*, vol. 32, no. 1, pp. 179–188, Jan. 2009.
[59] *PktGen, Pktgen Version 2.7.7 Using DPDK-1.7.1.* Accessed on Aug. 4, 2017. [Online]. Available: https://github.com/Pktgen/Pktgen-DPDK/

Michail-Alexandros Kourtis received the Diploma and master's degrees in computer science from the Athens University of Economics and Business in 2011 and 2014, respectively. He is currently pursuing the Ph.D. degree with the Department of Communications Engineering, University of the Basque Country (UPV/EHU), Bilbao, Spain. Since 2010, he has been with NCSR "Demokritos," in various research projects. His research interests include video processing, video quality assessment, image processing, network function virtualization, software defined networks, and quality of service.

Michael J. McGrath received the B.Sc. degree in analytical science, the Ph.D. degree in sensors and instrumentation, and the Graduate Diploma degree in information technology from Dublin City University, Dublin, Ireland, in 1992, 1995, and 1999, respectively, and the Graduate Diploma degree in computing and the M.S. degree in computing from the Institute of Technology Blanchardstown, Dublin, in 2004 and 2007, respectively. He is a Senior Researcher with Intel Labs Europe. He has been with Intel for 15 years, holding a variety of operational and research roles.

Georgios Gardikis received the Diploma degree in electrical and computer engineering and the Ph.D. degree from the National Technical University of Athens in 2000 and 2004, respectively. His expertise lies in the fields of digital terrestrial and satellite broadcasting, distribution networks for multimedia service provisioning, quality of experience assessment and novel mechanisms for application/network coupling. He is currently a Research and Development Manager with Space Hellas S.A.

Georgios Xilouris received the B.Sc. degree in physics from the University of Ioannina in 1999 and the M.Sc. degree in automation systems from the National Technical University of Athens in 2001. Since 2000, he has been a fellow researcher with the Institute of Informatics and Telecommunications, NCSR "Demokritos." He has participated in numerous EU-funded projects and was the Technical Coordinator of T-NOVA (FP7). He is currently a member of the Organising Committee for IEEE NFV-SDN.



Vincenzo Riccobene received the Diploma degree in computer engineering and the master's and Ph.D. degrees from the University of Catania in 2007, 2011, and 2016, respectively. He is currently a Research Engineer with Intel Labs Europe on research activities related to the orchestration of cloud computing environments, with a special focus on network function virtualization workloads. He was also involved in the activity of Intel Labs within the T-NOVA European FP7 project.

Panagiotis Papadimitriou received the Ph.D. degree in electrical and computer engineering from the Democritus University of Thrace, Greece, in 2008. He is currently on the faculty with the Department of Applied Informatics, University of Macedonia, Greece. From 2011 to 2016, he was an Assistant Professor with the Communications Technology Institute, Leibniz University Hannover, Germany. His research activities currently span next-generation Internet architectures, NFV, SDN, cloud networking, and mobile edge computing.

Eleni Trouva received the Diploma degree in computer engineering from the University of Patras, Greece, in 2006 and the Master of Science degree in computer science from the Department of Informatics, Athens University of Economics and Business in 2009. In 2013, she joined the Institute of Informatics and Telecommunications, NCSR "Demokritos." Her current research interests include future network architectures, NFV orchestration and management, and SDN and NFV security challenges.

Francesco Liberati received the master's degree in control engineering and the Ph.D. degree in systems engineering from the University of Rome "La Sapienza," Rome, Italy, in 2011 and 2015, respectively. From 2015 to 2017, he has been an Assistant Professor with eCampus University, Novedrate, Italy. His main research interests include smart grids, control of virtualized network infrastructures, and critical infrastructure protection.

Marco Trubian received the Ph.D. degree in electronic engineering in 1992, from the Politecnico di Milano, Italy. Since December 2002 he is an Associate Professor of Operational Research at the Computer Science Department of Universita' degli Studi di Milano. He has published in international journals such as *Networks, Discrete Applied Mathematics, Informs Journal on Computing*, the *European Journal of Operational Research*, the *Journal of Parallel and Distributed Computing*, and others.

Josep Batallé received the degree in telecommunications science from Universitat Rovira I Virgili and the Master of Science degree from Universitat Politecnica de Catalunya. Since 2012, he has been a Research and Development Engineer with i2CAT Foundation.

Harilaos Koumaras received the B.Sc. degree in physics from the Physics Department, University of Athens, in 2002, the M.Sc. degree in electronic automation and information systems in 2004, and the Ph.D. degree in computer science from the Computer Science Department, University of Athens, in 2007. He has been an active Research Associate with the Media Net Laboratory, National Centre of Scientific Research "Demokritos" since 2003.

David Dietrich (S'14–M'15) is a Post-Doctoral Researcher with the Institute of Communications Technology, Leibniz University Hannover, Germany. He participated in several research projects, e.g., the EU-funded T-NOVA project and the national project G-Lab. His research interests include network virtualization and network management with focus on algorithms and optimization. He is a member of ACM.

Aurora Ramos received the master's degree in telecommunication engineering in 2004 and Research Sufficiency in network engineering in 2007. She has been involved for over ten years in networks related research and development and innovation international projects for both academic and industry entities, having a deep knowledge of integration solutions for heterogeneous networks. Her current research work is focused in software networks. She has led the design and implementation of a novel NFV Marketplace within T-NOVA project.

Jordi Ferrer Riera received the degree in computer science from the Facultat d'Informàtica de Barcelona, Technical University of Catalonia (UPC) and the Ph.D. degree from the Design and Evaluation of Broadband Networks and Services Group, Telematics Engineering Department, UPC. He achieved the Telematic Networks and Operating Systems profile.

José Bonnet is a Senior Technology Consultant with AlticeLabs, where he has been applying his knowledge and expertise in developing carrier grade systems, gathered over a career of over 20 years, to NFV/SDN projects, preferably using Agile/DevOps methodologies and tools. He led the Orchestrator implementation work package in the EU FP7 T-NOVA project, as well as the equivalent work package in the EU Horizon2020/5GPPP SONATA Service Platform and Orchestration work package. He is currently part of the EU H2020 5Gtango Team.

Antonio Pietrabissa received the degree in electronic engineering and the Ph.D. degree in system engineering from the "Sapienza" University of Rome in 2000 and 2004, respectively, where he was a Post-Doctoral Researcher with the Department of System and Computer Science from 2004 to 2009 and has been an Assistant Professor since 2010 and currently holds the courses. He is scientific responsible for the research unit in two projects funded by the European Union (FP7 SWIPE, FP7 T-NOVA).

Alberto Ceselli is currently an Associate Professor with the Department of Computer Science, Università degli Studi di Milano. His activities include mathematical programming, design, and experimental analysis of algorithms for combinatorial optimization problems and prescriptive data analytics.

Alessandro Petrini received the undergraduation degree in applied mathematics from the University of Milan with a thesis on GPU-accelerated computation of PageRank, where he is currently pursuing the degree in computer science, involved in several parallel/high performance and GPU accelerated computing research projects in the field of bioinformatics. He is a full time collaborator of T-Nova project in both VNF mapping algorithms and VNF development.