Elastic Network Functions: Opportunities and Challenges

R. Szabo[1], M. Kind[2], F.-J. Westphal[2], H. Woesner[3], D. Jocha[1] and A. Csaszar[1]

[1]*Ericsson Research;* [2]*DT-AG;* [3]*BISDN*

Abstract

Network Function Virtualization (NFV) and Software Defined Networking (SDN) are key technology enablers for cost reductions and new business models in networking. The possibility to automatically and dynamically scale network services at run time is one of the main claims of NFV. Elastic NFV could be similar to what elastic cloud services provide for compute, with pay-per-use cost models for customers. However, control of resources for elastic services is far from trivial. We show how current NFV and SDN architectures could support elastic resource services for Network Functions (NFs). We reveal that the current NFV architecture does not allow recursive resource orchestration, therefore preventing resource scaling requests from being handled by a resource orchestrator overseeing the entire domain where an NF is executed. We introduce a logical centralization of joint compute and network resource orchestration as a UNIFY framework, which enables direct control of elastic resources for the NFs. We show opportunities and challenges associated to such an architecture.

I. Introduction

Socio-economic drivers, progress in information technologies, tumbling switching- and compute hardware costs and availability of open source software solutions are creating the conditions for a change of paradigm in designing and operating telecommunication networks and service infrastructures. Network Function Virtualization (NFV) [1] by the European Telecommunications Standards Institute (ETSI) and Software Defined Networking (SDN) [2] by the Open Networking Forum seem to be key technology enablers in the direction of meeting requirements such as cost reductions and new business models. NFV, on the one hand, targets at virtualizing servers and appliances that provide network functions. One of NFV's value propositions is cost optimization with the usage of common off the shelf hardware to reduce capital expenditure. This, together with increased operational efficiency is expected to also reduce operational expenditure. Operational efficiency is achieved by the automation of commission, configuration, resource management, etc. for even an order of magnitude higher number of managed elements than before. SDN, on the other hand, targets at breaking the vertical integration of network data and control planes to introduce (logically centralized) control plane programmability for novel networking virtualization (abstraction), simplified network (re)configuration and policy enforcement.

Today, all these promises are not yet realized. A typical example is the device located at the edge of carrier networks. Here fixed network operators require Broadband Network Gateway (BNG) functionality for customer centric processes like authentication, policing, etc. Technically, those gateways are physical devices hosting special network services or software combining network control and data plane aspects for fulfilling the processes, sometimes even supported by hardware acceleration. Given the fact, that the Broadband Network Gateway is operated like an appliance, it is difficult to add and evolve the service offering. For example, adding a parental control function or an Intrusion Detection System (IDS) for a certain customer is rather cumbersome. Today, this would require the instantiation of a separate IDS application, forwarding enforcement between the Network Gateway and the Intrusion Detection, and a policy interface to configure blocking of malicious traffic. Dynamism in changing control parameters like addresses adds additional management complexity.

The NFV framework [3] promises to remedy the problems of flexible service creation by managing and orchestrating softwarized network functions into telecommunication data centers. However, considering the full lifecycle of flexible network services, e.g., fulfillment, assurance and billing [4], one must look beyond the fulfillment phase and see that flexibility is also ensured during assurance (operation) phase.

We investigate, how network services can scale up/down or scale in/out to offer real elastic, pay as use services. Our main contribution is the analysis of opportunities and challenges related to dynamic service scaling with respect to i) the NFV framework; and ii) a novel and recursive resource orchestration framework.

In Sec. II we revisit the SDN and the NFV architectures revealing their synergies and differences; introduce a novel unifying architecture for joint and logically centralized compute and network orchestration. In Sec. III we discuss elastic services in the view of the different architectures. In Sec. IV we highlight related elasticity challenges and opportunities. Finally, we conclude in Sec. V.

II. Virtualization: SDN, NFV and Beyond

Open Networking Forum works on the definition of an SDN architecture [2]. They focus on three layers: data, control and application plane layers, but also include traditional management plane and end user systems into their architecture. SDN applications are defined as control plane functions operating over the forwarding abstraction offered by the SDN Controller. The applications connect to the SDN controller via the `A-CPI` reference point (see Fig. 1). The control plane's main responsibilities are i) creating the domain wide abstraction for internal use; ii) creating application or client SDN controller specific virtualization and policy enforcement; and iii) coordinating resources and operations for virtualization. The data plane of this architecture constitutes of Network Elements (NEs) that deal directly with customer traffic. These elements are connected to the controller. The right hand side of Fig. 1 shows an illustration of the SDN components and the corresponding reference points (see details in [2]).

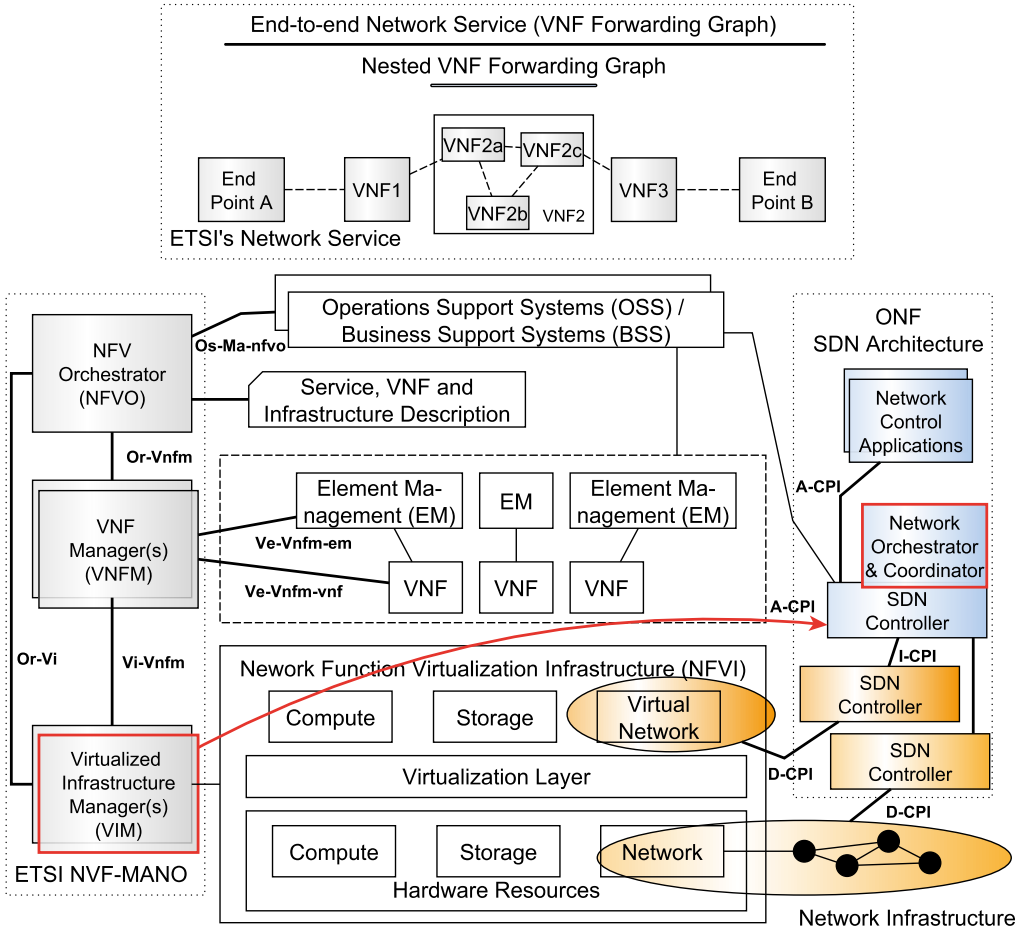

Fig. 1. The NFV and the SDN architectures side-by-side

Since the architecture allows other SDN Controllers (clients) to connect to the North of an SDN Controller the architecture is recursive. Therefore, automated network orchestration can be executed in multi-level virtualization environments, as long as resource virtualization and client policies related to resource use can be set. Such recursive automation enables clear separation of roles, responsibilities, information hiding and scalability. It also provides efficient operations in multi-technology, multi-vendor, multi-domain or multi-operator environments.

If we look into the user service aspects, then flexible service definition and creation may start by abstracting and formalizing the service into the concept of network service. In the NFV framework [3] the network service is formulated as a Virtualized Network Function (VNF) forwarding graph (see top of the Fig. 1). These graphs represent the way in which service end points (e.g., customer's access) are interconnected with the desired VNFs, such as firewalls, load balancers, DHCP servers, etc. Service graph representations form the input for the management and orchestration to instantiate and configure the requested service.

The main ETSI NFV Management and Orchestration (MANO) components are the Network Function Virtualization Orchestrator (NFVO) for the lifecycle management of the services; VNF Managers (VNFMs) for lifecycle management of individual VNFs; and Virtualized Infrastructure Managers (VIMs) for controlling and managing compute, storage and network resources [5] (see left side of Fig. 1). VNFs are instantiated in a NFV Infrastructure by the VIM through technology specific resource controllers like a SDN Controller or a Compute Controller. The orchestration framework also defines major reference points and is completed by connections to traditional management functions, which are also shown in Fig. 1 and detailed in [5].

If we put side-by-side the SDN and the NFV architectures, we can see that the VIM talks to an SDN Controller to orchestrate the virtualized network in the NFV Infrastructure (red arrow in Fig. 1). Logically, however, the VIM and the NFVO will perform network resource orchestration equivalently to a network orchestrator and coordinator within the SDN architecture (red box in the right hand side of Fig. 1).

There is, however, at least one major difference between the designs of the NFV and SDN architectures: SDN relies on a basic forwarding abstraction which can be reused recursively for virtualization of topology and forwarding elements, while the NFV framework offers significantly different services on the top compared to what is consumes at the bottom. Therefore, the current MANO framework is incapable of automated recursive orchestration for NFV.

A. Unifying compute and network virtualization (UNIFY)

Common open interfaces to programmatically control resources have been long pursued, e.g., [6], [7], but not yet realized. We believe that the hype around NFV allows us to reconsider a combined abstraction of compute and network resources. We propose to logically centralize all the resource orchestration functionalities existing distributively in the NFV MANO framework to enable automated recursive resource orchestration and domain virtualization similar to the SDN architecture. We show that the logical centralization of joint compute and network resource orchestration enables direct control of elastic resources for the network functions. This embedding of resource control within a deployed service resembles a recursive architecture.

In order to harmonize and logically centralize programmability over compute and network virtualization a common abstraction has to be defined [8]. In the framework of the UNIFY project [9], we combined compute and network resources to create a unified narrow waist programmability abstraction for Resource Orchestrator (RO). The UNIFY architecture is very similar to the SDN architecture, but it operates at the joint compute and network abstraction in each of its components. Fig. 2 shows the UNIFY architecture with its reference points.

The role of the Controller Adapter (CA) is to create a domain wide joint abstraction for the RO by interfacing with all possible infrastructure components. The `Ca-Co` reference point corresponds to compute or network controllers for compute or network only domains; agents directly managing compute

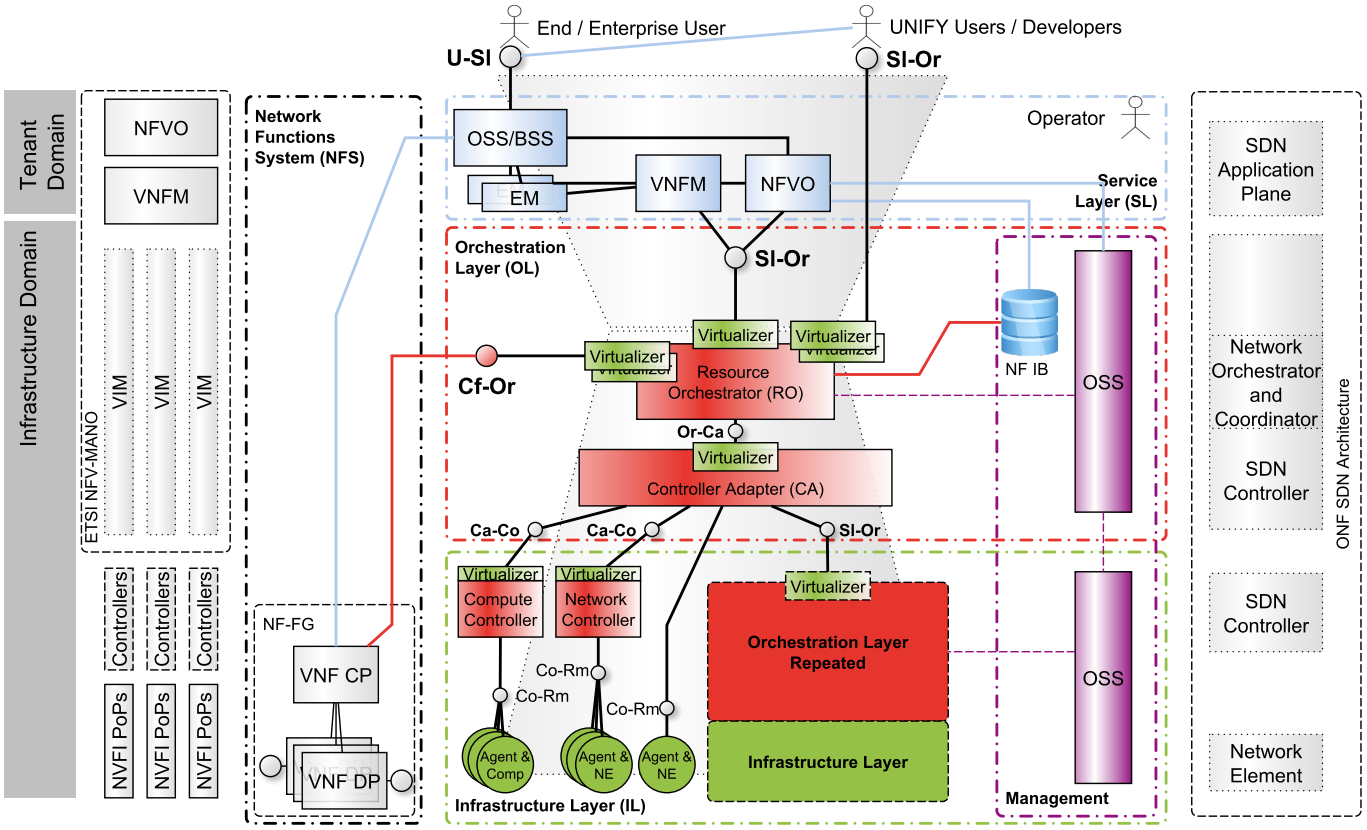

Fig. 2. Unifying Carrier and Cloud Resources (UNIFY)

or network resources are connected via a `Co-Rm` reference point, while other UNIFY domains use an `Sl-Or` reference point.

The RO offers resources orchestration services at the `Sl-Or` reference point with isolations based on requests and resource virtualization per client. UNIFY clients can be other UNIFY domains, the Service Provider itself or instantiated Network Functions (NFs). The resource orchestration services accepts Network Function Forwarding Graph (NF-FG) [9] requests, which contain abstract NF types; SDN forwarding rules between the NFs; and deployment constraints associated to arbitrary grouping of NFs. Additionally, the RO offers elastic control services at the `Cf-Or` reference point directly to NFs. Major differences between the `Cf-Or` and the `Sl-Or` reference points are: i) `Cf-Or` virtualizations must be created autonomously upon NF-FG instantiations, when detecting corresponding interface definitions at an NF's description (see port "C" in Fig. 3); ii) the connection between the NF and the `Cf-Or` interface must autonomously be inserted into the NF-FG request; and iii) the virtualization at the `Cf-Or` must be strictly scoped to the subset of the genuine NF-FG limiting the resource control to the NF's responsibility domain. For example, in Fig. 5 the `E-IDSC` can only control the `IDS` components but cannot control the `BNG VM`.

We further assume that the infrastructure management and the service management is separated (see purple Operation Support System (OSS) and the blue Element Management (EM) and OSS/Business Support System (BSS)). This way the UNIFY components are assumed to be managed from a dedicated infrastructure OSS representing the ownership of the domain wide virtualized resources.

In relation to the NFV and the SDN architectures, the RO replaces VIMs, and offers resource orchestration functions to VNFMs and the NFVO. Throughout the `Sl-Or` reference point, the UNIFY architecture resembles a recursive control and orchestration framework similar to the SDN. See left (NFV) and right (SDN) hand sides of Fig. 2.

III. Elastic Services

The possibility to dynamically scale network services at run-time in an automated fashion is one of the main advantages offered by the NFV approach, providing both better resource utilization and better service at a lower cost. Elastic NFV could be similar to what elastic cloud services provide for compute, with pay as you use cost models for the customers.

There are multiple reasons for initiating a scaling procedure: a user could request higher capacity ahead of time in order to deal with a known increase of demand in the future; the operations and management system could initiate scaling to maintain service level requirements based on monitoring Key Performance Indicators and resource use; or the deployed service components themselves could manage their resource needs similarly to automatic multi-threading scaling of some software running on multi-core CPUs.

Scaling of VNFs can be done in many ways. Which one is appropriate in a particular case depends heavily on the precise function provided by the VNFs, for example, the type of traffic it operates on and at which layer in the networking stack; requirements on state synchronization; requirements on control traffic during the scaling event; dependencies on other VNFs (service logic); ability to parallel processing or multi-threading; etc.

We analyze the capability of network services to self-adapt to changing conditions in (quasi) real-time in the next sub-sections.

A. Who is in charge of an elastic service?

According to the NFV MANO framework the VNFM is responsible for lifecycle management of individual VNFs. For example, the VNFM function may monitor Key Performance Indicators of a VNF to determine scaling operations. Scaling may include changing the configuration of virtualized resources, like adding/removing CPUs, adding/removing Virtual Machines (VMs) and adding/removing of associated network resources. VNFMs are assumed to be generic and exposed by an open interface for the VNFs (see Fig. 1). However, beyond resource control, VNFM's other functionalities are VNF instantiation and configuration; updates, upgrades and modifications; collection of VNF related performance measurements and event information; VNF assisted or automated healing; and coordination and adaptation between the Element Manager and the VIM.

With the UNIFY framework, we argue that the unified compute and network programmatic interface offered directly to VNFs at the `Cf-Or` reference point can enable VNF developers to autonomously control their resources needs. This means that elastic control may indeed be VNF specific and may be changed with VNF updates by the developers themselves. Adding this to the recursively hierarchical structure of the UNIFY architecture resource control requests may be handled locally or closest in the hierarchy to the actual execution unlike with a central management entity like the VNFM.

B. Monolithic vs. decomposed network functions: control and data plane split design

The SDN abstraction of the forwarding behavior enables the separation of traditional monolithic control and data plane network function designs. Such split may allow direct control of data plane resources and instances by the corresponding control functions. The MANO framework, however, does not address such split NFV designs, but assumes that such control is within the VNFM or NFVO.

For example, an IDS, whose role is to identify and block malicious traffic, could be implemented in various ways. Let's assume that the IDS service is managed by the operator, i.e., the Service Provider manages and operates the IDS upon the user's subscription to the service. The user's and the service's view of the IDS is shown in the left hand side of Fig. 3 including traditional management components. The IDS functional block can be realized as a hardware based monolithic component (a); a monolithic IDS VM (b); or as a data and control plane split design according to options (c) and (d).

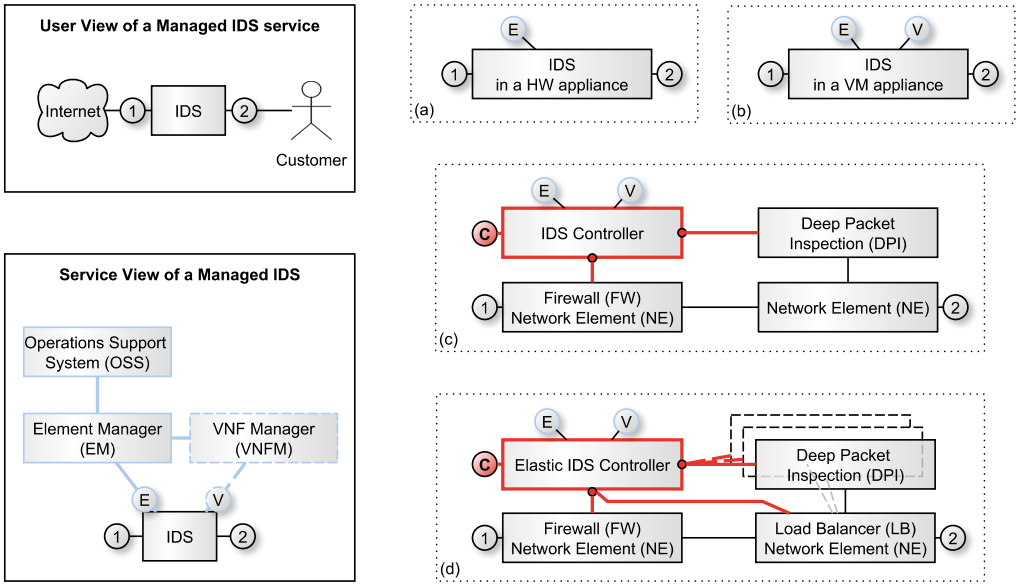

Fig. 3. IDS Decomposition Example

In option (a) one can consider only sharing of the hardware resources, in option (b) one can scale up/down the service dynamically by requesting or releasing resources associated to the individual components. However, scaling out/in requires adding to or removing from components and also re-configuring the service chain.

In option (c) the IDS control logic is separated into a Control VM, a firewall component for blocking of the malicious traffic and a traffic analyzer. The firewall may be mapped to a standard SDN network element and the traffic analysis may be realized by some generic Deep Packet Inspection (DPI) VM. Therefore, the Control VM must configure the generic DPI with patterns to identify malicious traffic.

In option (d), without loss of generality, we consider that the DPI processing can be parallelized by forking additional DPI instances. In this case, the Control VM must be extended to monitor the loads of the DPI components; to on-demand add or remove DPI instances and to re-configure traffic steering. For example, the SDN switch will not only mirror user traffic to a single DPI instance but must also act as a load balancer among the multiple DPI instances. Therefore, the load balancer must be dynamically configured by the IDS Controller. `Cf-Or` reference point of UNIFY offers such resource service for the IDS Control. By the means of NF-FG requests, the IDS Control can directly request adding or removing of DPI instances and the reconfiguration of the forwarding overlay. This is achieved without the NFV VNFM component (see Fig. $3(d)$).

Note also, that the virtualization between the `Cf-Or` and the IDS Control can be derived based on the VNF decomposition model presented in Fig. $3/(d)$. Basically, the existence of a control NF is detected by recognizing the specific `Cf-Or` interface (C interface in Fig. 3) and the scope of control is given by the components derived from the genuine service definition.

C. Example scenarios

In what follows, we analyze elastic control scenarios with the NFV (see Fig. 4) and the UNIFY (see Fig. 5) architectures. Fig. 4 and 5 show $i)$ the customer's view of the service (with dashed lines), $ii)$ the service provider's management view (Element Managements and the OSS), $iii)$ virtualization managers (NFVO, VNFM and VIM or RO) and $iv)$ the deployed network functions systems. The customer service is a cascade of an elastic IDS as shown in Fig. $3/(d)$ and a Broadband Network Gateway to connect the user to the Internet.

In Fig. 4 and 5 signaling messages are labeled and shown by arrow-heads for which we assume that there is always an underlying logical network connection. Also, the arrow-heads only illustrate our discussion of the primarily control message flow, but control messages may also pass in the reverse direction.

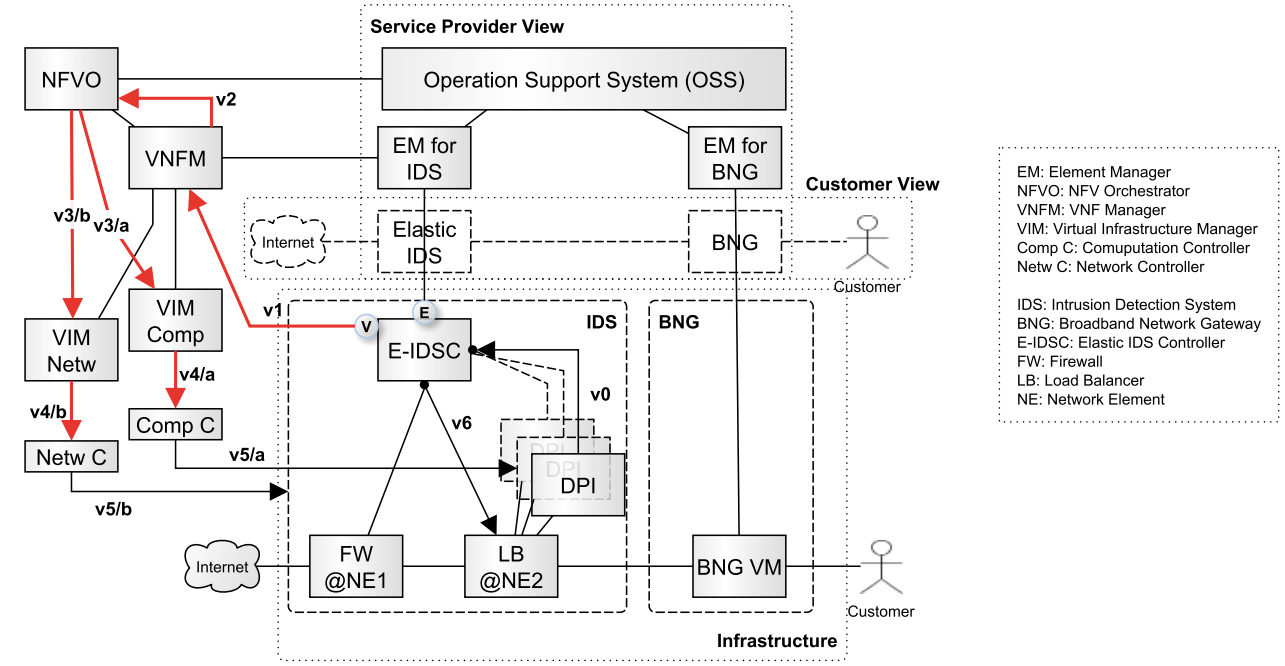

Fig. 4. Elastic control loop with the NFV framework

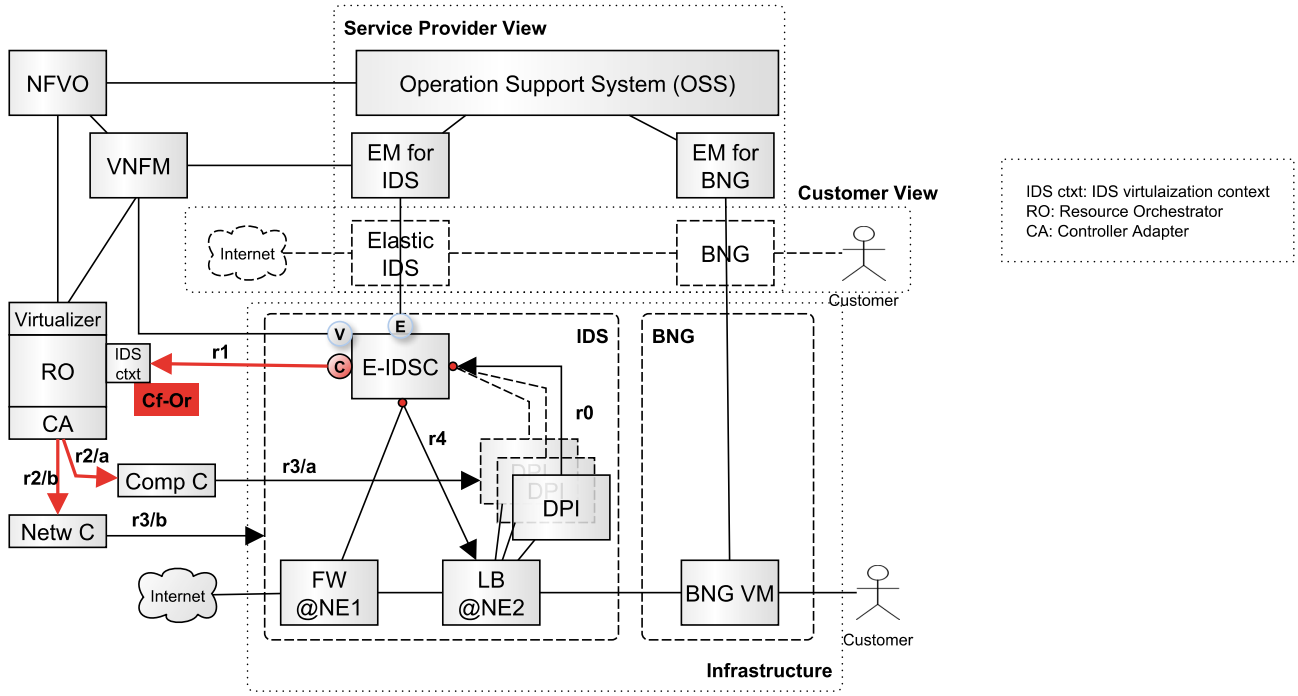

Fig. 5. Elastic control loop with the UNIFY framework

D. Service deployment with service decomposition

In the NFV approach, the NFVO must decide between the deployment options presented in Fig. 3 during orchestration. Afterwards, the NFVO sends a fully mapped VNF Forwarding Graph to the VIMs, who by the means of a compute (Comp C) and network (Netw C) controllers instantiate VNF VMs and configure the forwarding overlay respectively (see Fig. 4).

In the UNIFY case shown in Fig. 5, thanks to the logically centralized orchestrator (RO), the NFVO may send only the service provider view of the service graph to the RO. The RO, considering the VNF decomposition model as presented in Fig. 3, can decide, which option to initiate based on the available infrastructure resources. Furthermore, if another UNIFY domain is embedded under the RO (see repeated orchestration layer in Fig. 2), then the decomposition may be deferred to that sub-domain. In either cases, the services offered through different decomposition options must be equivalent; ROs shall consider service requirements and operational policies like energy efficiency, utilization, etc. when selecting between deployment options.

Additionally, during the UNIFY decomposition processes, ROs must detect if any VNF definition contains a Cf−Or interface (denoted by C in Fig. 3 and 5) interface. For each such a control interface $i)$ a new scoped virtualization must be created at the RO with an interface $ii)$ to which a logical connection from the VNF's C interface must be included into the forwarding overlay definition. In the example given in Fig. 5, the E−IDSC control VM is connected to the IDS ctxt virtualization context at the RO. The IDS ctxt visualization context contains only the components derived from the IDS service decomposition according to Fig. 3, hence excludes the BNG VM.

Note that the number of *external* interfaces related to a service component does not change during the decomposition process (see interfaces 1 and 2 in Fig. 3). The Cf−Or interface of a VNF is internally resolved to a connection to the corresponding RO. If there are multiple levels of virtualization domains, then the Cf−Or interface is resolved when first seen by an RO. The connection between such control interfaces and the corresponding RO is only included into the NF-FG description where it happens and does not propagate up in the hierarchy. This, thanks to the recursive orchestration framework, allows a UNIFY domain to re-optimize or decompose a VNF implementation without affecting external operations.

Once the corresponding RO is done with the decomposition and mapping of the service to the virtualized infrastructure resources it consults the Controller Adapter to instantiate and configure the components. For the embedded UNIFY domain, the Controller Adapter simply pass the corresponding sub service definition for the underlying orchestrator.

Last but not least, both in the NFV and the UNIFY cases, Element Managements and the OSS must handle any remaining service configurations.

E. Control of elasticity

In the following, we will analyze how the NFV and the UNIFY frameworks handle a scaling request for the example IDS service. We will refer to the control messages between the components with according to their labels in Fig. 4 and 5.

We assume that the increased load can only be met by an additional DPI instance. In both set-ups, the overload is learned by the E−IDSC by reports from the DPI components (message $v0$ in Fig. 4 and $r0$ in Fig. 5).

In the NFV setup, the VNFM is in charge of the scaling of the IDS service component. Therefore, the E−IDSC must notify the VNFM ($v1$) about the need of extra resources. Since the VNFM is service logic agnostic [5], it shall not know the decomposition details of the IDS service. Therefore it cannot know what further components to deploy, but must let the NFVO know ($v2$) about the problem. Note here, that even the scale up/down of a single DPI instance looks problematic, as the resources to be increased does not directly belong to the E−IDSC. The NFVO must look-up the service specific template to get an idea how to add more resources. Once it learns that a new DPI should be instantiated and connected into the service, it can allocate resources and configure the overlay by sending requests to a compute VIM (VIM

Comp) ($v3/a$) and to a network VIM (VIM Netw) ($v3/b$). VIM Comp can instantiate a new DPI via a compute controller, e.g., OpenStack Controller (see messages $v4/a, v5/a$). The network overlay must be configured similarly, via the VIM Netw and the SDN Controller (Netw C) (see messages $v4/b, v5/b$).

Note that the E-IDSC (VNF in general) must learn that the scaling operation was done, e.g., to be able to update the service logic. In our example, the E-IDSC may wish to reconfigure the load balancing rules in LB@NE2 ($v6$). Therefore, there must be a reverse flow of control information from the VIMs, through the NFVO, VNFMs to the requesting VNF.

In the UNIFY setup, the E-IDSC is directly connected to the RO (see Fig. 5). Let's assume again, that the E-IDSC learns that DPIs are running low with resources ($r0$). In this case, the E-IDSC knows the service logic inherently; it is designed and developed as part of it. It knows its instantiated VNFs, the corresponding forwarding overlay and an abstract view of the infrastructure resources through the RO's dedicated virtualizer. Therefore, the E-IDSC is in the best position to initiate scale up/down or scale out/in. The E-IDSC can issue an NF-FG request ($r1$), which can contain an additional DPI instance beyond the already deployed ones. In turn, the RO verifies the associated client policies, performs orchestration for the updated NF-FG and issues execution commands to compute and network domains (messages $r2/a$ and $r2/b$ respectively). The rest of the execution is identical to the NFV case beyond the two controllers.

Note that if the IDS components are allocated in a UNIFY sub-domain, then a hierarchy of ROs may be involved in the orchestration. However, if the decomposition happens only at the lowest level of the hierarchy, closest to the technology specific infrastructure domain, then orchestration at higher hierarchical levels may be left out of the process. This allows local decisions unless insufficient resources, in which case the execution might fall back to the NFV scenario.

As a summary, we show the major signaling messages of the NFV and the UNIFY set-ups as a comparison in Fig. 6, where components and messages labels correspond to Fig. 4 and 5. It is important to note, that not only the fewer elements, their potential "distance" to the executed NFs matters, but also the fact if a VNFM and NFVO can bear with service / application specific scaling details or not. In the UNIFY approach, such knowledge is placed into the NFs themselves.

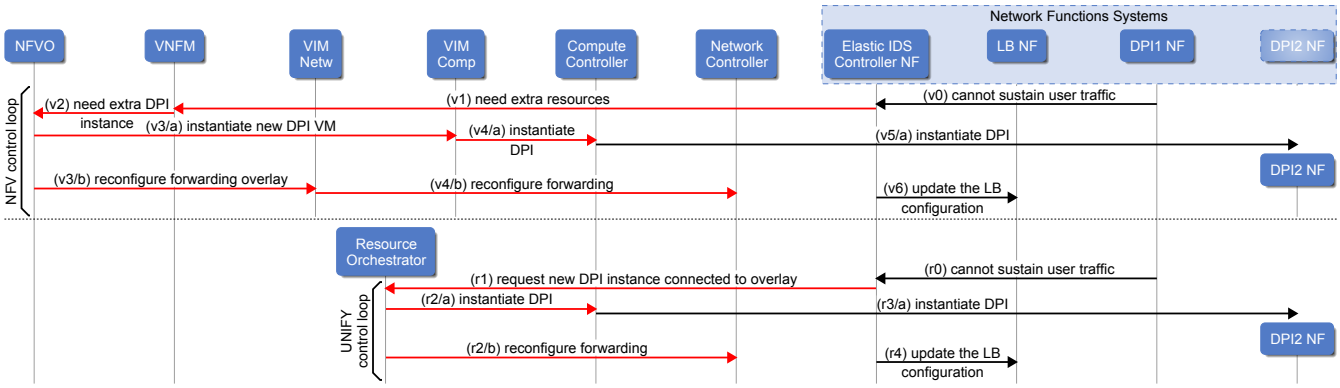

Fig. 6. Comparison of the NFV and UNIFY elastic control loops

IV. Challenges and Opportunities

First of all, service decomposition into atomic building block looks promising from component re-usability point of view (see, for example, DPI components in Fig. 3). Where and how decompositions happen affects virtualization related management components. In the NFV approach such decisions happen in the NFVO; in UNIFY we argue that decomposition and control of the corresponding sub-components are service specific, hence best placed in the hand of the service developers and realized within the control plane of the network functions.

Service decomposition can also imply control and data plane split designs. In such setups the control of data plane elasticities may rightfully be placed into the deployed control plane components instead of the overarching virtualization managers like the NFVO or VNFMs.

For control and data plane split deployments, appropriate control network overlays must also be created dynamically. These extra overlays should autonomously be added to the genuine service definition, for example, see options (c) and (d) in Fig. 3. Additionally, elastic control interfaces must also be autonomously connected (see the connection of the `Cf-Or` interface of the `E-IDSC` to the RO in Fig. 5). Therefore, a user's network function is connected to an infrastructure orchestrator, which, however, controls domain wide resources. Autonomous scoping of controllability and policy enforcement is a must at such elasticity control reference points.

How to learn about the new data plane structure after a scaling operation is another question. If decomposition and scaling operations are handled in the NFVO or VNFMs, then it is open how a control plane network function can learn about the revised data plane structure it oversees. If control plane network functions are in charge of their data plane scaling operations, they inherently know the results. This also motives our design choices behind the elasticity control services of the UNIFY framework.

Another issue is who should configure the data plane components, for example, the load balancer or the DPIs in Fig. 4 and 5? If scaling is non-transparent to the involved network functions (e.g., stateful processing), then a service logic must assist the operational reconfiguration of components. The question is, can an external virtualization management component (e.g., NFVO, VNFM or RO) bear with such a logic? We believe that such logic is best placed in control applications themselves. This is in harmony with the end-to-end principle of Internet: application specific functions should reside in the application (e.g., control plane network functions), rather than within the network (e.g., virtualization managers).

Regarding security issues, in the NFV setup elasticity control signaling goes through trusted elements like Element Managements, the OSS, VNFMs, the NFVO and VIMs. Therefore, it is relatively easy to enforce user policies related to the service contract due to the involvement of service layer management functions. In the case of the UNIFY approach, however, resource policies related to service contracts must be enforced in corresponding virtualizers distributed in the hierarchy of UNIFY domains. However, UNIFY orchestrators do not have any notion of service logics but only compute, storage and network abstractions. Therefore, configurations for virtualizers and policy enforcements must autonomously be translated from service logics. For example, the virtualizer and the policy enforcement associated to the `E-IDSC` in the RO (see Fig. 5) could limit

- the available resources (max 10 virtual CPU, max 5 Gbyte RAM, less than 100Mbps in/out rate, maximum cost of resources, etc.),
- the capabilities as available VNF types (DPI and network elements) for the control application.

We believe, that such policy information can be extracted from genuine service requests and client policies (e.g., maximum cost, etc.).

V. Summary

The importance of control and elasticity capabilities for the Future Internet has already been pointed out, e.g., in [10]. The recent NFV initiation with SDN may be able to provide elastic networking services similar to elastic cloud services in compute. We have discussed the synergies and differences between the SDN and the NFV architectures. We have shown that the ETSI NFV MANO framework does not use unified compute and network abstractions for automated resource orchestration. This prevents recursion and automation, which affects who could be in charge for controlling and providing elasticity of services. We have shown that the UNIFY architecture has appropriate resource abstractions for recursive automation of resource orchestrations, yielding for possible outplacement of elasticity control to deployed network functions systems. Throughout the analysis of an example scenario, we discussed opportunities and challenges related to both, the NFV and the UNIFY management and orchestration frameworks.

We conclude with some highlighted opportunities considering the UNIFY approach:

- faster resource control loops with less involved elements;
- clearer separation of roles and responsibilities;
- logical centralization of all resource orchestration functions;
- recursion and automation of resource orchestration across multi-level virtualization domains;
- enabler for split control and data plane design;
- resource driven service decomposition;
- *interfacing directly with deployed network functions for (3rd party) elastic control;*

and associated challenges:

- autonomous service decomposition for control and data plane split designs;
- autonomous definition of resource virtualizations and policy enforcements for deployed control functions;
- autonomous creation of management and control network overlays.

Initial proof of concept validations for the UNIFY orchestration framework had been showcased at SIGCOMM 2014 [11] and at the European Workshop on Software Defined Networking (EWSDN) 2014 [12]. The orchestration framework is released as an open source [13]. We hope to trigger further discussions in this hot topic area with our ideas.

ACKNOWLEDGMENT

We are grateful for the invaluable comments and suggestions to the anonymous reviewers.

This work is supported by FP7 UNIFY, a research project partially funded by the European Community under the Seventh Framework Program (grant agreement no. 619609). The views expressed here are those of the authors only. The European Commission is not liable for any use that may be made of the information in this document. The authors are thankful to the UNIFY team for the inspiring discussions.

REFERENCES

[1] ETSI, "White Paper: Network Functions Virtualisation (NFV)," 2013. [Online]. Available: http://portal.etsi.org/NFV/NFV_White_Paper2.pdf

[2] Open Networking Forum (ONF), "SDN architecture," ONF, TR (Technical Reference), non-normative, type 2 SDN ARCH 1.0 06062014, Jun. 2014. [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/TR_SDN_ARCH_1.0_06062014.pdf

[3] ETSI, "Architectural framework," ETSI, Group Specification v1.1.1, Oct. 2013. [Online]. Available: http://docbox.etsi.org/ISG/NFV/Open/Published/gs_NFV002v010101p%20-%20Architectural%20Fwk.pdf

[4] TM Forum, "Business Process Framework (eTOM)," online, Feb. 2015. [Online]. Available: http://www.tmforum.org/BusinessProcessFramework

[5] Jrgen Quittek et al., "Network function virtualization (NFV) management and orchestration," ETSI, GROUP SPECIFICATION GS NFV-MAN 001 V1.1.1, Nov. 2014. [Online]. Available: http://www.etsi.org/deliver/etsi_gs/NFV-MAN/001_099/001/01.01.01_60/gs_NFV-MAN001v010101p.pdf

[6] A. Palo, L. Zuccaro, A. Simeoni *et al.*, "A common open interface to programmatically control and supervise open networks in the future internet," in *Future Network and Mobile Summit (FutureNetworkSummit), 2013*, Jul. 2013, pp. 1–9.

[7] M. Castrucci, M. Cecchi, F. Delli Priscoli *et al.*, "Key concepts for the future internet architecture," in *Future Network Mobile Summit (FutureNetw), 2011*, Jun. 2011, pp. 1–10.

[8] P. Skoldstrom, B. Sonkoly, A. Gulyas *et al.*, "Towards unified programmability of cloud and carrier infrastructure," in *2014 Third European Workshop on Software-Defined Networks (EWSDN)*, http://ewsdn.eu, 2014.

[9] R. Szabo, B. Sonkoly, M. Kind *et al.*, "D2.2: Final Architecture," UNIFY Project, Deliverable 2.2, Nov. 2014. [Online]. Available: http://fp7-unify.eu/files/fp7-unify-eu-docs/Results/Deliverables/UNIFY%20Deliverable%202.2%20Final%20Architecture.pdf

[10] A. Galis, H. Abramowicz, M. Brunner *et al.*, "Management and service-aware networking architectures (MANA) for future internet; position paper: System functions, capabilities and requirements," in *Fourth International Conference on Communications and Networking in China, 2009. ChinaCOM 2009*, Aug. 2009, pp. 1–13.

[11] A. Csoma, B. Sonkoly, L. Csikor *et al.*, "ESCAPE: Extensible Service ChAin Prototyping Environment using Mininet, Click, NETCONF and POX. Demonstation." in *ACM SIGCOMM 2014*, 2014.

[12] ——, "Multi-layered service orchestration in a multi-domain network environment. Demonstation." in *2014 Third European Workshop on Software-Defined Networks (EWSDN)*, http://ewsdn.eu, 2014.

[13] UNIFY Project, "Extensible Service ChAin Prototyping Environment using Mininet, Click, NETCONF and POX," open source, Sep. 2014. [Online]. Available: https://github.com/nemethf/escape