Internet Research Task Force NFVRG
Internet-Draft
Intended status: Informational
Expires: May 20, 2016

G. Carrozzo, Ed.
Nextworks
R. Szabo, Ed.
Ericsson
K. Pentikousis, Ed.
EICT

November 17, 2015

Network Function Virtualization: Resource Orchestration Challenges
draft-caszpe-nfvrg-orchestration-challenges-00

Abstract

Network function virtualization (NFV) promises improved operations in terms of flexibility, efficiency, and manageability, but orchestration, in general, and recursive orchestration, in particular, is still an item of ongoing research. We summarize the current state of the art in open-source initiatives in this area and present current directions of research and development in terms of orchestration, resource decomposition and federation, policy-based resource management, measurement and analytics, and virtual network function (VNF) elasticity.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

To a large degree there is agreement in the network research, practitioner, and standardization communities that rigid network control limits the flexibility and manageability of service creation, as discussed in [NSC] and the references therein. Today operators run IP networks stitched together by a plethora of highly specialized middleboxes that implement firewall, NAT, DPI, traffic scrubbing, and other functionality [middlebox]. In this environment orchestration is fragmented, hindering rapid service deployment. Moreover, recursive orchestration for heterogeneous resources is not practically employed. In this memo, we examine orchestration support in current open-source efforts and describe research challenges in this area.

1.1. Motivation

Flexible service definition and creation start by formalizing the service into the concept of network function forwarding graphs, such as the ETSI VNF Forwarding Graph [ETSI-NFV-Arch] or the ongoing work in IETF [RFC7498]. These graphs represent the way in which service end-points (e.g., customer access) are interconnected with a set of selected network functionalities such as firewalls, load balancers, and so on, to deliver a network service. Service graph representations form the input for the management and orchestration to instantiate and configure the requested service. For example, ETSI defined a Management and Orchestration (MANO) framework in [ETSI-NFV-MANO]. We note that throughout such a management and orchestration framework different abstractions may appear for separation of concerns, roles or functionality, or for information hiding.

In the simplest case, technology specific domains (e.g., software or networking) create a technology specific abstraction and control interfaces (e.g., Open Stack Controller or an SDN Controller), which are connected to an overarching orchestrator. Figure 1 shows an NFVO orchestrating three NFVI PoPs: two DCs with OpenStack Controller (OSC) and a Wide Area Network domain by an SDN Controller.

```
                             +---------+
                             |  NVFO   |
                             +---------+
                           .---/  |  \---.
                          /       |       \
                     +------+   +------+   +------+
                     | VIM |    | WIM |    | VIM |
                     |(OSC)|    |(SDN)|    |(OSC)|
                     +------+   +------+   +------+
```

Figure 1

However, technology specific separation of the orchestration interface is not necessarily rational once the orchestrator (e.g., NFVO) aggregated all the different resource types into a joint abstraction. Figure 2 shows two autonomous systems with their own technology specific resource domains plus a resource virtualization between the two domains. In Figure 2 RO denotes resource orchestrators, Service O denotes service lifecycle management and VIM/WIM denote technology specific resource managers. Further, for the interworking interface between the two autonomous domains the resource virtualization is denoted by S (as a slice). Control over

the virtualized resource offered by RO 1 to RO 2 is assumed to be exercised also by the API exposed at S. Assume that the resource virtualization contains both i) networking resources (e.g., topology of forwarding elements) and ii) software resources capable of hosting VNFs. S is interworking interface, which needs standardization for efficient interworking.

```
              * +---------+
              * |Service O|
              * +---------+
              *     |
              * +---------+
              * |   RO 2  |
              * +-[aggr-]-+
               *   | \ \
                *  |  \ \----------.
                 * |   \----.       \
    +---------+   *|     +--+--+   +-----+
    |Service O|    |*    | WIM |   | VIM |
    +---------+    | *   |(SDN)|   |(OSC)|
             |     |  *  +-----+   +-----+
           +------[S]+ *             AS 2
           |   RO 1  |  ******************
           +-[aggr-]-+               AS 1
         .---/  |  \---.
        /       |       \
   +-----+   +-----+   +-----+
   | VIM |   | WIM |   | VIM |
   |(OSC)|   |(SDN)|   |(OSC)|
   +-----+   +-----+   +-----+
```

Figure 2

As soon as resources become part of multiple administration domains; technology or vendor specific domains or multi-operator setup standardized interfaces are key for smooth interworking. Without such interoperability different technologies for data center and network operation result in distinct technology domains even within a single carrier. Multi technology/domain barriers start to emerge hindering the full programmability of the NFVI and limiting the potential for rapid service deployment.

1.2. Scope

In this document we consider the virtualization and resource orchestration of heterogeneous resources and functions over multiple technology, vendor or administrative domains.

Examples of application contexts for recursive orchestration include, but are not limited to, the following:

- large scale experimentation over software networks, based on slices of network and compute resources from different federated providers;

- operators who intend to implement their network service offer over virtual infrastructure in the form of virtual network and compute resources and functions, all procured as a service from physical infrastructure providers.

1.3. Terminology

We use the term software, compute and "compute and storage" interchangeably throughout the document. Moreover, we use the following definitions, some of which are established in [ETSI-NFV-Arch] and are copied here for reader convenience.

Network Function Virtualization (NFV)
The principle of separating network functions from the hardware they run on by using virtual hardware abstraction.

NFV Infrastructure Point of Presence (NFVI PoP)
Any combination of virtualized compute, storage and network resources.

NFV Infrastructure (NFVI)
Collection of NFVI PoPs under one orchestrator.

Virtual Network Function (VNF)
One or more virtual machines running different software and processes on top of industry-standard high-volume servers, switches and storage, or cloud computing infrastructure, and capable of implementing network functions traditionally implemented via custom hardware appliances and middleboxes (e.g. router, NAT, firewall, load balancer, etc.)

Virtualized Network Function Forwarding Graph (VNF   FG)
An ordered list of VNFs creating a service chain.

VNF Island

A set of virtualized network functions and related network and compute resources under the same administrative ownership/control. A VNF island could consist of multiple zones, each characterized by a specific set of control tools and interfaces.

VNF Zone

A set of virtual network functions grouped for homogeneity of technologies and/or control tools and/or interfaces (e.g. L2 switching zone, optical switching zone, OpenFlow controlled zone, other transit domain zone with a control interface). The major goal of defining SDN zones is to implement appropriate policies for increasing availability, scalability and control of the different resources of the island. Examples of zone definitions are available in popular cloud management systems like CloudStack (e.g. refer to the CloudStack Infrastructure partitioning into regions, zones, pods, etc., [cloudstack]) and OpenStack (e.g. refer to the infrastructure partitioning in availability zones and host aggregates [openstack]).

Transit network domains

Network domains can use a bandwidth-on-demand interface to expose automatically and on-demand control of connectivity services and, optionally, inter-domain topology exchange. In order to federate resources of distant facilities (i.e. islands/zones) it must be ensured that interconnectivity is provided on-demand and with a specific granularity.

Slice

A provider-created subset of virtual networking and compute resources, created from physical or virtual resources available for the (slice) provider.

Resource Orchestrator (RO)

Entity responsible for domain wide global orchestration of network services and software resource reservations in terms of network functions over the physical or virtual resources the RO owns. The domain an RO oversees may consist of slices of other domains.

Resource Manager (RM)

Entity responsible for controlling and managing different types of resources and/or network functions.

Management and Orchestration (MANO)

In the ETSI NFV framework [ETSI-NFV-MANO], MANO is the global entity responsible for management and orchestration of NFV lifecycle.

Further, we make use of the following terms:

NF is a network function, either software-based (VNF) or appliance-based.

SW is a (routing/switching) network element with a programmable control plane interface.

DC is a data center network element which in addition to a programmable control plane interface offers a DC control interface.

LS is a logical switch instance.

CN is an element equipped with compute and/or storage resources.

UN or universal node, is a network element that integrates and manages in a unified platform both compute and networking components.

2. Examples

2.1. DC - WAN - DC Example

Assume that the set of virtual network and non-network functions is determined, reserved and deployed across the different islands, the resulting virtual network environment is ready for being used by any tool or application the user wants to deploy in it. Figure 3 illustrates a resource orchestrator (RO) as a functional entity whose task is to map the service graph to the infrastructure resources under some service constraints and taking into account the NF resource descriptions.

```
                                        |[Service Graph]
                                        V
                                +--------------------+
                                |Service Orchestrator|
                                +--------------------+
                                        |
                                        V
              ^    +--------+   +--------------------+
              |    | NF-IB  |   |Resource            |
              |C   |Resource|<->|Orchestrator (RO0)  |
              |O   | Descr. |   |                    |
              |N   +--------+   +---[aggregator]-----+
              |T                      A A A
              |R         .-----------/  |  \---------.
              |O         |              |            |
              |L       +-S-+          +-S-+        +-S-+
              |        |RO1|          |RO2|        |RO3|
              V        +---+          +---+        +---+

              ^                       +---+
              |D                   +--|SW3|--+
              |A                   |  +---+  |
              |T       +---+       |         |      +---+
              |A    1  |PoP|     +---+     +---+    |PoP|  8
              |     o--|DC1|-----|SW2|-----|SW4|----|DC2|--o
              V        +---+     +---+     +---+    +---+

                   [-  SP1  -][-       SP2      -][-  SP3  -]
```

Figure 3: Resource Orchestrator: network function information base (NF-IB), inputs and output

NF resource descriptions in the NF-IB are assumed to contain information necessary to map NF types to a choice of instantiable VNF flavor or a selection of an already deployed NF appliance and networking demands for different operational policies. For example, if energy efficiency is to be considered during the decision process then information related to energy consumption of different NF flavors under different conditions (e.g., network load) should be included in the resource description.

Note that we also introduce a new service provider (SP0) which effectively operates on top of the virtualized infrastructure offered by service providers SP1, SP2 and SP3.

In order for the RO to execute the resource mapping it needs to operate on the combined control plane illustrated in Figure 4. In

this figure we mark clearly that the interfaces to the compute (DC) control plane and the SDN (SW) control plane are distinct and implemented through different interfaces/APIs. For example, Ic1 could be the Apache CloudStack API, while Ic2 could be a control plane protocol such as ForCES or OpenFlow [RFC7426]. In this case, the orchestrator at SP0 (top part of the figure) needs to maintain a tight coordination across this range of interfaces.

```
                               +---------+
                               |Orchestr.|
                               |   RO0   |
                          _____+---------+_____
                         /          |          \
                        /           V Ic2       \
                       |       +---------+       |
                   Ic1 V       |SDN Ctrl |       V  Ic3
              +---------+      |   RO2   |      +---------+
              |Comp Ctrl|      +---------+      |Comp Ctrl|
              |   RO1   |        /  |  \        |   RO3   |
              +---------+    +---   V   ----+   +---------+
                   |         |    +----+    |         |
                   |         |    |SW3 |    |         |
                   V         |    +----+    |         V
                  +----+     V   /      \   V     +----+
               1  |PoP |    +----+      +----+    |PoP |  8
               o--|DC1 |----|SW2 |------|SW4 |----|DC2 |--o
                  +----+    +----+      +----+    +----+

              [-   SP1  -][-        SP2       -][-  SP3   -]
              [-                    SP0                   -]
```

Figure 4: The RO Control Plane view. Control plane interfaces are indicated with (line) arrows. Data plane connections are indicated with simple lines.

2.2. Monitoring and AAA

Figure 5 illustrates Resource Orchestrators (RO) which are responsible for orchestrating end-to-end network services and resources reservations in the whole infrastructure. Moreover, ROs should be able to delegate end-to-end resource and service provisioning in technology-agnostic way.

ROs are connected to different types of Resource Managers (RMs), which are in turn used to control and manage different kinds of technological resources. For example, the RM (WAN) provides

connectivity between L1/L2 network domains at the two ends. This management can be achieved using frame, packet or circuit switching technologies and should support different protocols.

On the other hand, for instance, the RM (LAN) manages the network infrastructure composed of SDN-enabled devices, e.g. OpenFlow switches or routers. In short, it can control the user traffic environment by updating flow tables in physical devices.

AAA is a cross layer function facilitating authN/authZ procedures in federated facilities. Similarly, monitoring allows to retrieve, correlate and abstract statistics from the different components of the physical and virtual resources.

```
   +---+  +------------------------------------------+  +---+
   |   |  |          Resource Orchestrator - RO      |  |   |
   |   |--|                                          |--|   |
   |   |  +------------------------------------------+  |   |
   |   |                    ...                  ...    |   |
   |   |                     |                    |     |   |
   |   |                     |                    |     |   |
   | M |  +-------------------------------+     +----+  |   |
   | O |--|            RO                 | ... | RO |--| A |
   | N |  +-------------------------------+     +----+  | A |
   | I |        |           |         |           |     | A |
   | T |        |           |         |           |     |   |
   | O |  +-----------+ +-------+ +-------+     +----+  |   |
   | R |  |  Resource | |  RM   | |  RM   |     | RM |  |   |
   | I |--|  Manager  | |       | |       |     |    |--|   |
   | N |  | (compute) | | (LAN) | | (WAN) |     |    |  |   |
   | G |  +-----------+ +-------+ +-------+     +----+  |   |
   |   |        |           |         |            |    |   |
   |   |  +-----------+ +-------+ +-------+     +----+  |   |
   |   |--| NVFI PoP  | |Network| |Network|     |NFVI|--|   |
   +---+  +-----------+ +-------+ +-------+     +----+  +---+
```

Figure 5: Recursive Orchestration: Monitoring and AAA

3. Review of Open Orchestration Frameworks

3.1. OpenStack

Among cloud orchestration solutions, OpenStack is the de facto common reference through its Heat module [os-heat]. OpenStack Heat implements an orchestration engine to launch multiple composite cloud applications based on templates in the form of text files that can be

treated like code. Many existing CloudFormation templates can be launched on OpenStack. Heat provides both an OpenStack-native REST API and a CloudFormation-compatible Query API.

A Heat template describes the infrastructure for a cloud application in a text file. Infrastructure resources that can be described include: servers, floating IPs, volumes, security groups, users, etc. Templates can also specify the relationships between resources (e.g. this volume is connected to this server). Heat also provides an auto-scaling service.

Heat primarily manages cloud infrastructure, does not support federation and AAA is bundled in the OpenStack framework.

3.2. OpenMANO

OpenMANO is an open source project which implements the reference architecture for Management and Orchestration under standardization at ETSI NFV ISG (NFV MANO) [openmano]. OpenMANO consists of two main SW components:

o NFV VIM (Virtualised Infrastructure Manager) to provide computing and networking capabilities and to deploy virtual machines.

o A reference implementation of an Network Functions Virtualisation Orchestrator (NFV-O), which allows the creation and deletion of VNF templates, VNF instances, network service templates and network service instances.

OpenMANO does not support federation and AAA as of today.

3.3. UNIFYing Carrier Network and Cloud Resources (UNIFY)

The UNIFY project pursues full network and service virtualization to enable rich and flexible services and operational efficiency. The main goal of the project is to unify software and network resources in a common framework. The UNIFY API combines compute, storage and network resources into a joint programmatic reference point, allowing multi-level virtualization and orchestration of Network Function Forwarding Graph for fast and flexible service chaining. While the ambition is similar to ETSI NFV, the project pursues a recurring control architecture, which similar to the ONF's SDN control plane architecture [ONF-SDN-ARCH]. The UNIFY's problem statement and challenges are documented in [I-D.unify-nfvrg-challenges] while the proposed virtualization and control API is defined in [I-D.unify-nfvrg-recursive-programming].

3.4. Federated Experimentation Infrastructures

The FELIX project implements federation and integration of different network and compute resources controlled via SDN and Network Service Interface (NSI) in a multi-domain heterogeneous environment across, initially spanning Europe and Japan. The FELIX architecture extends and advances assets previously developed (e.g. in OFELIA), by realizing the federation concepts defined in SFA [SFA] with a combination of recursive and hierarchical orchestration, request delegation and inter-domain dependency management. Further details are available in [I-D.felix-nfvrg-recursive-orchestration] and the references therein.

4. Challenges

Orchestrating networking resources appears to have a recursive nature at different levels of the hierarchy. Would a programmatic interface at the combined compute and network abstraction better support this recursive and constraint-based resource allocation? Further, can such a joint compute, storage and network programmatic interface allow an automated resource orchestration similar to the recursive SDN architecture [RFC7426][ONF-SDN-ARCH]? Below we we summarize key questions and challenges which arise from the recursive resource orchestration and management concepts.

4.1. Resource description

A prerequisite for joint placement decisions of compute, storage and network is the adequate description of available resources. There have been manifold attempts to create frameworks for resource description, most prominently RDF of W3C, NDL, the GENI RPC and its concept of Aggregate Managers, ONF's TTP and many more. Quite naturally, all attempts to standardize "arbitrary" resource descriptions lead to creating ontologies, complex graphs describing relations of terms to each other.

Practical descriptions of compute resources are currently focusing on number of logical CPU cores, available RAM and storage, allowing, e.g., the OpenStack Nova scheduler to meet placement decisions. In heterogeneous network and compute environments, hardware may have different acceleration capabilities (e.g., AES-NI or hardware random number generators), so the notion of logical compute cores is not expressive enough. In addition, the network interfaces (and link load) provide important information on how fast a certain VNF can be executed in one node.

This may lead to a description of resources as VNF-FGs themselves. Networking resource (SW) may expose the capability to forward and

process frames in, e.g., an OpenFlow TableFeatures reply. Compute nodes in the VNF-FG would expose lists of capabilities like the presence of AES hardware acceleration, Intel DPDK support, or complex functions like a running web server. An essential part of the compute node's capability would be the ability to run a certain VNF of type X within a certain QoS spec. As the QoS is service specific, it can only be exposed by a control function within the instantiated VNF-FG.

4.2. Dependencies (de-composition)

Salt [SALT], Puppet [PUPPET], Chef [CHEF] and Ansible [ANSIBLE] are tools to manage large scale installations of virtual machines in DC environments. Essentially, the decomposition of a complex function into its dependencies is encoded in "recipes" (Chef).

The OASIS TOSCA [TOSCA] specification aims at describing application layer services to automate interoperable deployment in alternative cloud environments. The TOSCA specification "provides a language to describe service components and their relationships using a service topology".

Is there a dependency (decomposition) abstraction suitable to drive resource orchestration between application layer descriptions (like TOSCA) and cloud specific installations (like Chef recipes)?

4.3. Elastic VNF

In many use cases, a VNF may not be designed for scaling up/down, as scaling up/down may require a restart of the VNF which the state data may be lost. Normally a VNF may be capable for scaling in/out only. Such VNF is designed running on top of a small VM and grouped as a pool of one VNF function. VNF scaling may cross multiple NFVI PoPs (or data center)s in order to avoid limitation of the NVFI capability. At cross-DC scaling, the result is that the new VNF instance may be placed at a remote cloud location. At VNF scaling, it is a must requirement to provide the same level of Service Level Agreement (SLA) including performance, reliability and security.

In general, a VNF is part of a VNF Forwarding Graph (VNF FG), meaning the data traffic may traverse multiple stateful and stateless VNF functions in sequence. When some VNF instances of a given service function chain are placed / scaled out in a distant cloud execution, the service traffic may have to traverse multiple VNF instances which are located in multiple physical locations. In the worst case, the data traffic may ping-pong between multiple physical locations. Therefore it is important to take the whole service function chain's performance into consideration when placing and scaling one of its

VNF instance. Network and cloud resources need mutual considerations, see [I-D.zu-nfvrg-elasticity-vnf].

4.4. Measurement and analytics

Programmable, dynamic, and elastic VNF deployment requires that the Resource Orchestrator (RO) entities obtain timely information about the actual operational conditions between different locations where VNFs can be placed. Scaling VNFs in/out/up/down, VNF execution migration and VNF mobility, as well as right-sizing the VNFI resource allocations is a research area that is expected to grow in the coming years as mechanisms, heuristics, and measurement and analytics frameworks are developed.

For example, Veitch et al. [IAF] point out that NFV deployment will "present network operators with significant implementation challenges". They look into the problems arising from the lack of proper tools for testing and diagnostics and explore the use of embedded instrumentation. They find that in certain scenarios fine-tuning resource allocation based on instrumentation can lead to at least 50% reduction in compute provisioning. In this context, three categories emerge where more research is needed.

First, in the compute domain, performance analysis will need to evolve significantly from the current "safety factor" mentality which has served well carriers in the dedicated, hardware-based appliances era. In the emerging softwarized deployments, VNFI will require new tools for planning, testing, and reliability assurance. Meirosu et al. [I-D.unify-nfvrg-devops] describe in detail the challenges in this area with respect to verification, testing, troubleshooting and observability.

Second, in the network domain, performance measurement and analysis will play a key role in determining the scope and range of VNF distribution across the resources available. For example, IETF has worked on the standardization of IP performance metrics for years. The Two-Way Active Measurement Protocol (TWAMP) could be employed, for instance, to capture the actual operational state of the network prior to making RO decisions. TWAMP management, however, still lacks a standardized and programmable management and configuration data model [I-D.cmzrjp-ippm-twamp-yang]. We expect that as VNFI programmability gathers interest from network carriers several IETF protocols will be revisited in order to bring them up to date with respect to the current operational requirements. To this end, NFVRG can play an active role in identifying future IETF standardization directions.

Third, non-technical considerations which relate to business aspects or priorities need to be modeled and codified so that ROs can take intelligent decisions. Meirosu et al. [I-D.unify-nfvrg-devops] identify two aspects of this problem, namely a) how high-level network goals are translated into low-level configuration commands; and b) monitoring functions that go beyond measuring simple metrics such as delay or packet loss. Energy efficiency and cost, for example, can steer NFV placement. In NFVI deployments operational practices such as follow-the-sun will be considered as earlier research in the data center context implies.

5. IANA Considerations

No IANA considerations are applicable.

6. Security Considerations

This document does not have any impact on the security of the Internet.

7. Acknowledgements

This work has been partially supported and funded by the European Commission through the FP7 UNIFY (grant agreement no. 619609), FP7 ICT FELIX (grant agreement no. 608638) projects and the National Institute of Information and Communications Technology (NICT) in Japan. The views expressed here are those of the authors only. The European Commission and NICT are not liable for any use that may be made of the information in this document.

8. Contributors

The editors would like to acknowledge in alphabetical order the following contributors from the FELIX and UNIFY projects who have provided text, comments, pointers, and ideas during the development of earlier versions of this document: Bartosz Belter, Carlos Bermudo, Andras Csaszar, Diego Daino, Mario Kind, Tomohiro Kudoh, Catalin Meirosu, Zu Qiang, Jin Tanaka, Fritz-Joachim Westphal, and Hagen Woesner.

9. Informative References

[ANSIBLE] Ansible Inc., "Ansible Documentation", 2015, <http://docs.ansible.com/index.html>.

[CHEF] Chef Software Inc., "An Overview of Chef", 2015, <https://docs.chef.io/chef_overview.html>.

[cloudstack]
"Apache CloudStack documentation. Cloud Infrastructure Concepts", Available online at http://cloudstack.apache.org/docs/en-US/Apache_CloudStack/4.1.0/html/Admin_Guide/cloud-infrastructure-concepts.html.

[ETSI-NFV-Arch]
ETSI, "Architectural Framework v1.1.1", Oct 2013, <http://www.etsi.org/deliver/etsi_gs/NFV/001_099/002/01.01.01_60/gs_NFV002v010101p.pdf>.

[ETSI-NFV-MANO]
ETSI, "Network Function Virtualization (NFV) Management and Orchestration V0.6.1", Jul. 2014, <http://docbox.etsi.org/ISG/NFV/Open/Latest_Drafts/NFV-MAN001v061-%20management%20and%20orchestration.pdf>.

[FELIX-D2.1]
R. Krzywania, W. Bogacki, B. Belter, K. Pentikousis, T. Rothe, G. Carrozzo, N. Ciulli, C. Bermudo, T. Kudoh, A. Takefusa, J. Tanaka and B. Puype, , "FELIX Deliverable D2.1 - Experiment Use Cases and Requirements", Available online at http://www.ict-felix.eu., September 2013.

[FELIX-D2.2]
R. Krzywania, W. Bogacki, B. Belter, K. Pentikousis, T. Rothe, M. Broadbent, G. Carrozzo, N. Ciulli, R. Monno, C. Bermudo, A. Vico, C. Fernandez, T. Kudoh, A. Takefusa, J. Tanaka and B. Puype, , "FELIX Deliverable D2.2 - General Architecture and Functional Blocks", Available online at http://www.ict-felix.eu., February 2014.

[I-D.cmzrjp-ippm-twamp-yang]
Civil, R., Morton, A., Zheng, L., Rahman, R., Jethanandani, M., and K. Pentikousis, "Two-Way Active Measurement Protocol (TWAMP) Data Model", draft-cmzrjp-ippm-twamp-yang-02 (work in progress), October 2015.

[I-D.felix-nfvrg-recursive-orchestration]
Carrozzo, G. and K. Pentikousis, "Recursive orchestration of federated virtual network functions", draft-felix-nfvrg-recursive-orchestration-00 (work in progress), July 2015.

[I-D.unify-nfvrg-challenges]
Szabo, R., Csaszar, A., Pentikousis, K., Kind, M., Daino, D., Qiang, Z., and H. Woesner, "Unifying Carrier and Cloud Networks: Problem Statement and Challenges", draft-unify-nfvrg-challenges-02 (work in progress), July 2015.

[I-D.unify-nfvrg-devops]
Meirosu, C., Manzalini, A., Steinert, R., Marchetto, G., Papafili, I., Pentikousis, K., and S. Wright, "DevOps for Software-Defined Telecom Infrastructures", draft-unify-nfvrg-devops-03 (work in progress), October 2015.

[I-D.unify-nfvrg-recursive-programming]
Szabo, R., Qiang, Z., and M. Kind, "Towards recursive virtualization and programming for network and cloud resources", draft-unify-nfvrg-recursive-programming-02 (work in progress), October 2015.

[I-D.zu-nfvrg-elasticity-vnf]
Qiang, Z. and R. Szabo, "Elasticity VNF", draft-zu-nfvrg-elasticity-vnf-01 (work in progress), March 2015.

[IAF] Veitch, P., McGrath, M. J., and Bayon, V., "An Instrumentation and Analytics Framework for Optimal and Robust NFV Deployment", Communications Magazine, vol. 53, no. 2 IEEE, February 2015.

[middlebox]
A. Greenhalgh, F. Huici, M. Hoerdt, P. Papadimitriou, M. Handley, and L. Mathy, , "Flow Processing and the Rise of Commodity Network Hardware", ACM SIGCOMM Computer Communication Review Volume 39 issue 2, April 2009.

[NSC] John, W., Pentikousis, K., et al., "Research directions in network service chaining", Proc. SDN for Future Networks and Services (SDN4FNS), Trento, Italy IEEE, November 2013.

[ONF-SDN-ARCH]
ONF, "SDN architecture", Jun. 2014, <https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/TR_SDN_ARCH_1.0_06062014.pdf>.

[openmano]
"OpenMANO", Available online at https://github.com/nfvlabs/openmano/wiki.

[openstack]
"Scaling Openstack", Available online at http://docs.openstack.org/openstack-ops/content/scaling.html.

[os-heat] "OpenStack Orchestration - Heat", Available online at https://wiki.openstack.org/wiki/Heat.

[PUPPET] Puppet Labs., "Puppet 3.7 Reference Manual", 2015, <http://docs.puppetlabs.com/puppet/3.7/reference/>.

[RFC7426] Haleplidis, E., Ed., Pentikousis, K., Ed., Denazis, S., Hadi Salim, J., Meyer, D., and O. Koufopavlou, "Software-Defined Networking (SDN): Layers and Architecture Terminology", RFC 7426, DOI 10.17487/RFC7426, January 2015, <http://www.rfc-editor.org/info/rfc7426>.

[RFC7498] Quinn, P., Ed. and T. Nadeau, Ed., "Problem Statement for Service Function Chaining", RFC 7498, DOI 10.17487/RFC7498, April 2015, <http://www.rfc-editor.org/info/rfc7498>.

[SALT] SaltStack, "Salt (Documentation)", 2015, <http://docs.saltstack.com/en/latest/contents.html>.

[SFA] L. Peterson, R. Ricci, A. Falk and J. Chase, , "Slice-based Federation Architecture (SFA) v2.0", , July 2010.

[TOSCA] OASIS Standard, "Topology and Orchestration Specification for Cloud Applications Version 1.0", November 2013, <http://docs.oasis-open.org/tosca/TOSCA/v1.0/os/TOSCA-v1.0-os.html>.

Authors' Addresses

Gino Carrozzo (editor)
Nextworks
via Livornese 1027
Pisa 56122
Italy

Email: g.carrozzo@nextworks.it
URI: http://www.nextworks.it/

Robert Szabo (editor)
Ericsson Research, Hungary
Irinyi Jozsef u. 4-20
Budapest 1117
Hungary

Email: robert.szabo@ericsson.com
URI: http://www.ericsson.com/

Kostas Pentikousis (editor)
EICT
Torgauer Strasse 12-15
Berlin 10829
Germany

Email: k.pentikousis@eict.de
URI: http://www.eict.de