Internet Research Task Force (IRTF)
Internet Draft
Category: Experimental

R. Krishnan
Dell
N. Figueira
Brocade
Dilip Krishnaswamy
IBM Research
D. R. Lopez
Telefonica I+D
Steven Wright
AT&T
Tim Hinrichs
Styra
Ruby Krishnaswamy
Orange
Arun Yerra
Dell

Expires: March 2016 March 2, 2016

NFVIaaS Architectural Framework for Policy Based Resource Placement and Scheduling

draft-krishnan-nfvrg-policy-based-rm-nfviaas-06

Abstract

One of the goals of Network Functions Virtualization (NFV) is to offer the NFV infrastructure as a service to other SP customers – this is called NFVIaaS. Virtual Network Function (VNF) deployment in this paradigm will drive support for unique placement policies, given VNF's stringent service level specifications (SLS) required by customer SPs. Additionally, NFV DCs often have capacity, energy and other constraints – thus, optimizing the overall resource usage based on policy is an important part of the overall solution. The purpose of this document is to depict an architectural framework for policy based resource placement and scheduling in the context of NFVIaaS.

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that

other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire in March 2016.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document.

Conventions used in this document

Table of Contents

1. Introduction

One of the goals of NFV [ETSI-NFV-WHITE] is to offer the NFV infrastructure as a service to other SP customers - this is called NFVIaaS [ETSI-NFV-USE-CASES]. In this context, it may be desirable for a Service Provider to run virtual network elements (e.g., virtual routers, virtual firewalls, and etc. - these are called Virtual Network Functions - VNF) as virtual machine instances inside the infrastructure of another Service Provider. In this document, we call the former a customer SP and the latter an NFVIaaS SP.

There are many reasons for a customer SP to require the services of an NFVIaaS SP, including: to meet performance requirements (e.g., latency or throughput) in locations where the customer SP does not have physical data center presence, to allow for expanded customer reach, regulatory requirements, and etc.

As VNFs are virtual machines, their deployment in such NFVIaaS SPs would share some of the same placement restrictions (i.e., placement policies) as those intended for Cloud Services. However, VNF deployment will drive support for unique placement policies, given VNF's stringent service level specifications (SLS) required/imposed by customer SPs. Additionally, NFV DCs or NFV PoPs [ETSI-NFV-TERM] often have capacity, energy and other constraints - thus, optimizing the overall resource usage based on policy is an important part of the overall solution.

The purpose of this document is to depict an architectural framework for policy based resource placement and scheduling in the context of NFVIaaS.

2. NFVIaaS Architectural Framework for Policy Based Resource Placement and Scheduling

The policy engine performs policy-based resource placement and scheduling of Virtual Machines (VMs) in support for NFVIaaS. It determines optimized placement and scheduling choices based on the constraints specified in the policy. The NFVIaaS Architectural Framework for Policy Based resource placement and scheduling is based on the NFV policy architectural framework [IRTF-NFV-POLICY-ARCH].This is depicted in Figure 1.

In one instantiation of this architecture, the policy engine would interface with the Measurement Collector to periodically retrieve instantaneous per-server CPU utilization, which it would then use to compute a table of per-server average CPU utilization. In an alternative instantiation of this architecture, the measurement collector could itself compute per-server average CPU utilization. The latter approach reduces overhead, since it avoids too frequent pulling of stats from Ceilometer. The policy engine evaluates such policies based on an event trigger or a based on a programmable timer.

Other average utilization parameters such as VM CPU utilization, VM Memory utilization, VM disk read IOPS, Network utilization/latency etc. could be used by the policy engine to enforce other types of placement policies.

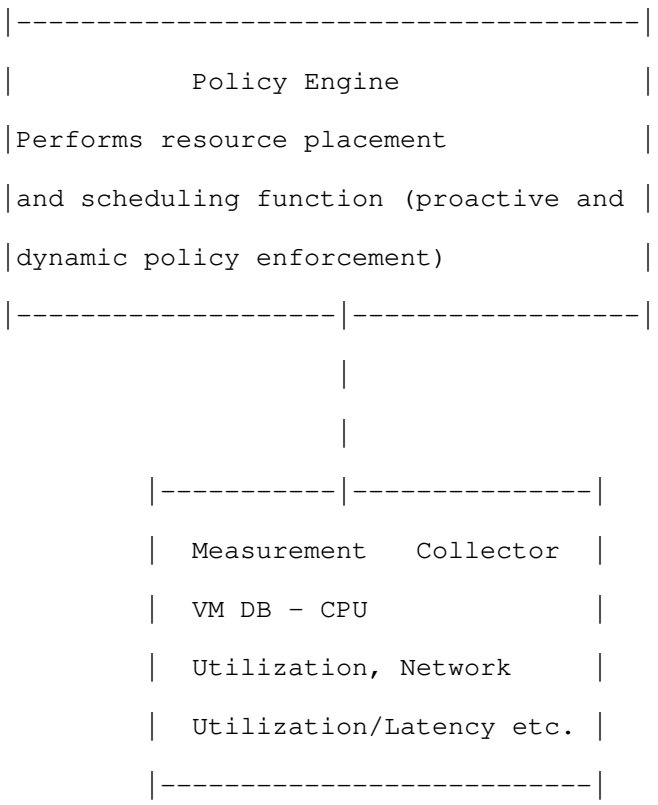


Figure 1: NFVIaaS Architecture for Policy Based Resource Placement and Scheduling

In an ETSI NFV Architectural Framework [ETSI-NFV-ARCH][NFV-MANO-SPEC], Policy Engine is part of the Orchestrator and the Measurement Collector is part of the Virtual Infrastructure Manager (VIM).

3. System Analysis in a OpenStack Framework

3.1. Compute Monitoring with OpenStack - Use Case and Example

Consider an NFVIaaS SP that owns a multitude of mini NFV data centers managed by OpenStack [OPENSTACK] where,

- The Policy Engine function is performed by OpenStack Congress [OPENSTACK-CONGRESS-POLICY-ENGINE]

- The Measurement Collector function is performed by OpenStack Celiometer [OPENSTACK-CELIOMETER-MEASUREMENT]

- The Policy Engine has access to the OpenStack Nova database that stores records of mapping of virtual machines to physical servers.

Exemplary Mini NFV DC configuration:

An exemplary mini NFV DC configuration is depicted below.

- There are 210 physical servers in 2U rack server configuration spread over 10 racks.

- There are 4 types of physical servers each with a different system configuration and from a particular manufacturer. It is possible that the servers are from the same or different manufacturer. For the purpose of this example, server type 1 is described further. Server type 1 has 32 virtual CPUs and 128GB DRAM from manufacturer x. Assume 55 physical servers of type 1 per mini NFV DC.

- There are 2 types of instances large.2 and large.3, which are described in the table below. Each parameter has a minimum guarantee and a maximum usage limit.

Instance Type	Virtual CPU Units Minimum Guarantee /Maximum Usage	Memory (GB) Minimum Guarantee /Maximum Usage	Minimum/Maximum Physical Server Utilization (%)

large.2	0/4	0/16	0/12.5
large.3	0/8	0/32	0/25

Table 1: NFVIaaS Instance Types

For the purpose of this example, the Mini NFV DC topology is considered static -- the above topology, including the network interconnection, is available through a simple file-based interface.

Policy 1 (an exemplary NFV policy):

Policy 1 is an exemplary NFV policy. In a descriptive language, Policy 1 is as follows - "For physical servers of type 1, there can be at most only one active physical server with average overall utilization less than 50%."

The goal of this policy is to address the energy efficiency requirements described in the ETSI NFV Virtualization Requirements [ETSI-NFV-REQ].

Policy 1 is an example of reactive enforcement.

Policy 2 (another exemplary NFV policy):

Policy 2 is necessary to protect NFV servers from failures. In this example we consider failures of physical servers. Policy 2 is as follows - "Not more than one VM of the same HA group must be deployed on the same physical server".

Note: There may be conditions (according to current Mini DC usage and policies of type 2 that are currently active) when there may not be any placement solution respecting both policies. It may be better to reformulate Policy 1? For example, "Minimize the number of physical servers with average overall utilization less than 50%"

Policy 2 is an example of proactive and reactive enforcement.
Various Module Interactions for Policy 1:

The various module interactions with respect the architectural framework in Figure 1 for Policy 1 is described below.

The policy calls for the identification of servers by type. OpenStack Congress would need to support server type, average CPU utilization, and be able to support additional performance parameters (in the future) to support additional types of placement policies. OpenStack Congress would run the policy periodically or based on events such as deleting/adding VMs etc. Initially, we could use a periodic timer based approach. In case OpenStack Congress detects a violation, it determines optimized placement and scheduling choices so that the policy is not violated.

OpenStack Congress could interface with OpenStack Celiometer to periodically retrieve instantaneous per-server CPU utilization, which it would then use to compute a table of per-server average CPU utilization. Alternatively, OpenStack Celiometer could itself compute per-server average CPU utilization which could be used by OpenStack Congress.

The proposed module interactions in this NFVIaaS placement policy example are as depicted in the architectural framework in Figure 1.

A key goal of Policy 1 above is to ensure that servers are not kept under low utilization, since servers have a non-linear power profile and exhibit relatively higher power wastage at lower utilization. For example, in the active idle state as much as 30% of peak power is consumed. At the physical server level, instantaneous energy consumption can be accurately measured through IPMI standard. At a customer instance level, instantaneous energy consumption can be approximately measured using an overall utilization metric, which is a combination of CPU utilization, memory usage, I/O usage, and network usage. Hence, the policy is written in terms of overall utilization and not power usage.

The following example combines Policy 1 and Policy 2.

For an exemplary maximum usage scenario, 53 physical servers could be under peak utilization (100%), 1 server (server-a) could be under partial utilization (62.5%) with 2 instances of type large.3 and 1 instance of type large.2 (this instance is referred as large.2.X1), and 1 server (server-b) could be under partial utilization (37.5%) with 3 instances of type large.2. Call these three instances large.2.X2, large.2.Y and large.2.Z

One HA-group has been configured and two large.2 instances belong to this HA-group. To enforce Policy 2 large.2.X1 and large2.X2 that belong to the HA-group have been deployed in different physical servers, one in server-a and a second in server-b.

When one of the large.3 instances mapped to server-a is deleted from physical server type 1, Policy 1 will be violated, since the overall utilization of server-a falls to 37,5%, since two servers are underutilized (below 50%)

OpenStack Congress, on detecting the policy violation, uses various constraint based placement techniques to find the new placement(s) for physical server type 1 to address Policy 1 violation without breaking Policy 2. Constrained based placement will be explored in a convex optimization framework [CONVEX-OPT]; some of the algorithms which would be considered are linear programming [LINEAR-PROGRAM], branch and bound [BRANCH-AND-BOUND], interior point methods, equality constrained minimization, non-linear optimization etc.

Various new placement(s) are described below.

1) New placement 1: Move 2 of three instances of large.2 running on server-b to server-a. Overall utilization of server-a - 62,5%. Overall utilization of server-b - 25%. large.2.X2 must not be one of the migrated instances.

2) New placement 2: Move 1 instance of large.3 to server-b. Overall utilization of server-a - 12,5%. Overall utilization of server-b - 62.5%.

A third solution consisting of moving 3 large.2 instances to server-a cannot be adopted since this breaks Policy 2. Another policy minimizing the number of migrations could allow choosing between solution (1) and (2).

New placements 2 and 3 could be considered optimal, since they achieve maximal bin packing and open up the door for turning off server-a or server-b and maximizing energy efficiency.

To detect violations of Policy 1, an example of a classification rule is expressed below in Datalog, the policy language used by OpenStack Congress.

Database table exported by the Resource Placement and Scheduler for Policy 1 and Policy 2:

The database table exported by the Resource Placement and Scheduler for Policy 1 is below.

server_utilization (physical_server, overall_util)

- Each database entry has the physical server and the calculated average overall utilization.

vm_host_mapping(vm, server)

- Each database entry gives the physical server on which VM is deployed.

anti-affinity_group(vm, group)

- Each entry gives the anti-affinity group to which a VM belongs.

Policy 1 (in Datalog [DATALOG] policy language):

Policy 1 in a Datalog policy language is as follows.

error (physical_server) :-

nova [OPENSTACK-NOVA-COMPUTE]: node (physical_server, "type1"),

resource placement and scheduler: server_utilization
(physical_server, overall_util < 50)

Policy 2 (in Datalog policy language):

error(vm) :-

anti-affinity_group(vm1, grp1),

anti-affinity_group(vm2, grp2),

grp1 != grp2,

nova: vm host mapping(vm1, server-1),

nova: vm host mapping(vm2,server-2),

server-1 == server-2

3.2. Joint Network and Compute Awareness with OpenStack - Use Case

There are several NFV DCs such as mobile base stations, small central offices, small branch office locations etc. Having an OpenStack Controller in each of these locations increases the management complexity and the DC capacity needs. The idea is to have the OpenStack compute and network nodes in these DC locations and manage them centrally through an OpenStack Controller node through a

larger central office location in the same metro area. The key considerations here are that the OpenStack management network extends over the metropolitan area network (MAN) and is an in-band network where the data and management traffic use the same physical pipe; typical MAN distances are within 100 miles but could have a geographic variation.

Across MAN, some of the key management network considerations are latency and bandwidth impacting various operations such as configuration, VM image transfer, monitoring data collection, backing up of logs etc. OpenStack Neutron can provide the hooks for reporting MAN bandwidth and latency which will be abstracted by the enhanced OpenStack Nova scheduler. The network connection between NFV DCs can be modelled as Neutron end-points. During the validation phase, the enhanced OpenStack Nova scheduler (part of the OpenStack Controller node) in the central DC can determine if the compute/network nodes in the small DC are manageable remotely based on the service SLA requirements specified by the Orchestrator; each of the small DCs could have different bandwidth/latency depending on the network topology. During runtime, the enhanced OpenStack Nova scheduler periodically monitors the MAN bandwidth and latency variations and reports any exceptions. The enhanced scheduler can measure the overall bandwidth usage across MAN so that it can determine the optimized time window(s) during the day for backing up logs, detailed monitoring data etc. The enhanced scheduler can also measure overall latency usage across MAN so that it can place the high availability instances across DCs meeting the service SLA requirements. Dynamic measurement of MAN bandwidth and latency can be implemented using a vendor specific OpenStack Neutron plugin which typically interfaces with an SDN Controller [LAYERED-SDN]. [Y.1731-monitoring] is typically used by switches for performance monitoring across physical/logical Ethernet network end points.

4. Related Work

A related proof of concept in ETSI NFV on placement and scheduling is [ETSI-NFV-POC-PLACEMENT].

5. Summary

6. Future Work

TBD

7. IANA Considerations

This draft does not have any IANA considerations.

8. Security Considerations

9. Contributors

Hesham ElBakoury

Huawei

Hesham.ElBakoury@huawei.com

10. Acknowledgements

The authors would like thank Prabhakar Kudva and Gokul Kandiraju for the proof of concept effort.

11. References

11.1. Normative References

11.2. Informative References

[ETSI-NFV-WHITE] "ETSI NFV White Paper,"
http://portal.etsi.org/NFV/NFV_White_Paper.pdf

[ETSI-NFV-USE-CASES] "ETSI NFV Use Cases,"
http://www.etsi.org/deliver/etsi_gs/NFV/001_099/001/01.01.01_60/gs_NFV001v010101p.pdf

[ETSI-NFV-REQ] "ETSI NFV Virtualization Requirements,"
http://www.etsi.org/deliver/etsi_gs/NFV/001_099/004/01.01.01_60/gs_NFV004v010101p.pdf

[ETSI-NFV-ARCH] "ETSI NFV Architectural Framework,"
http://www.etsi.org/deliver/etsi_gs/NFV/001_099/002/01.01.01_60/gs_NFV002v010101p.pdf

[ETSI-NFV-TERM] "ETSI NFV: Terminology for main concepts in NFV,"
http://www.etsi.org/deliver/etsi_gs/NFV/001_099/003/01.02.01_60/gs_NFV003v010201p.pdf

[OPENSTACK] "OpenStack Open Source Software,"
https://www.openstack.org/

[OPENSTACK-CONGRESS-POLICY-ENGINE] "A policy as a service open source project in OpenStack,"
https://wiki.openstack.org/wiki/Congress

[OPENSTACK-CELIOMETER-MEASUREMENT] "OpenStack Celiometer," http://docs.openstack.org/developer/ceilometer/measurements.html

[OPENSTACK-NOVA-COMPUTE] "OpenStack Nova," https://wiki.openstack.org/wiki/Nova

[LINEAR-PROGRAM] Dmitris Alevras and Manfred W. Padberg, "Linear Optimization and Extensions: Problems and Solutions," Universitext, Springer-Verlag, 2001.

[BRANCH-AND-BOUND] "Fundamentals of Algorithmics," G. Brassard and P. Bratley.

[NFV-MANO-SPEC] "NFV Management and Orchestration Framework Specification," http://docbox.etsi.org/ISG/NFV/Open/Latest_Drafts/NFV-MAN001v061-%20management%20and%20orchestration.pdf

[CONVEX-OPT] "Convex Optimization," https://web.stanford.edu/~boyd/cvxbook/bv_cvxbook.pdf

[ETSI-NFV-POC-PLACEMENT] "ETSI NFV Proof of Concept on Placement and Scheduling," http://nfvwiki.etsi.org/index.php?title=Constraint_based_Placement_and_Scheduling_for_NFV/Cloud_Systems

[DATALOG] Ceri, S. et al., "What you always wanted to know about Datalog (and never dared to ask)," Knowledge and Data Engineering, IEEE Transactions on (Volume:1 , Issue: 1)

[IRTF-NFV-POLICY-ARCH] Figueira, N. et al., "Policy Architecture and Framework for NFV and Cloud Services,"

[Y.1731-monitoring] "OAM functions and mechanisms for Ethernet-based networks," https://www.itu.int/rec/T-REC-Y.1731/en

[LAYERED-SDN] "Cooperating Layered Architecture for SDN," https://datatracker.ietf.org/doc/draft-contreras-sdnrg-layered-sdn/

Authors' Addresses

Ram (Ramki) Krishnan
Dell Inc.
ramkri123@gmail.com

Norival Figueira
Brocade Communications

nfigueir@Brocade.com

Dilip Krishnaswamy
IBM Research
dilikris@in.ibm.com

Diego Lopez
Telefonica I+D
Don Ramon de la Cruz, 82
Madrid, 28006, Spain
+34 913 129 041
diego.r.lopez@telefonica.com

Steven Wright
AT&T
sw3588@att.com

Tim Hinrichs
Styra
tim@styra.com

Ruby Krishnaswamy
Orange
ruby.krishnaswamy@orange.com

Arun Yerra
Dell Inc.
arun.yerra@dell.com

Internet Research Task Force (IRTF)
Internet Draft
Category: Informational

R. Krishnan
Brocade
Dilip Krishnaswamy
IBM Research
D. R. Lopez
Telefonica I+D
Asif Qamar
Evolv
Steven Wright
AT&T
Norival Figueira
Brocade

Expires: April 2015 November 11, 2014

NFV Real-time Analytics and Orchestration: Use Cases and Architectural Framework

draft-krishnan-nfvrg-real-time-analytics-orch-01

Abstract

One of the key goals of NFV is to optimize the infrastructure resource usage while driving operational simplicity. Real-time analytics providing insight into various components such as compute (e.g. dynamic CPU utilization), storage (e.g. dynamic capacity usage), network (e.g. dynamic bandwidth utilization), energy (e.g. dynamic power consumption) is key to not only providing visibility into the NFV infrastructure and thus driving operational simplicity but also optimizing resource usage for the purposes of orchestration. This draft focusses on use cases and architectural framework for real-time analytics and orchestration including Big Data predictive analytics for addressing the aforementioned requirements.

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire in April 2015.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document.

Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

Table of Contents

1. Introduction

Operator Network Function Virtualization Infrastructure Point-of-Presence (NFVI-PoP) locations [ETSI-NFV-TERM] often have capacity, energy and other constraints. Thus, optimizing overall resource usage is an important requirement [ETSI-NFV-REQ]. The general case must consider a distributed (elastic) VNF NFVI platform implementation where VMs running for different VNFs (with different characteristics) can co-exist in the same physical server. This case must address the goal of optimizing overall resource usage through mechanisms like bin-packing [BIN-PACK].

In this context, some of the important challenges faced are:

- Performance issues due to noisy neighbor effect, where a VM running for a VNF can affect the VM(s) running for another VNF.
- Security issues, especially due to inconsistent configuration in a dynamic environment where one VNF could affect others.
- Energy Efficiency given that servers have substantial idle power usage.
- Resources used (Compute, bandwidth, storage) for the real-time analytics in comparison to the VNF payload resource usage.

The purpose of this document is two-fold. First, it intends to discuss various use cases to describe the above challenges. Second, it will depict an architectural framework for real-time analytics and orchestration, applicable to the above use cases in a multi-vendor environment.

For the purposes of real-time analytics for orchestration, various metrics need to be collected, stored and analyzed.

Metrics collection: Metric collection may occur at different periods during the lifecycle of the VNF. Metric collection during an onboarding process in a controlled load configuration may provide a baseline for characterization of "normal" operational performance. Such baseline characterizations may be useful for detection "out of normal" performance at a later point in the VNF lifecycle.

Metrics storage: It is recommended to store and analyze metrics locally to minimize the costs of backhaul to remote locations.

Metrics analysis: The assumption here is that the metrics to be collected for analysis would be VNF independent in the sense that they would apply regardless of the type of VNF. Metrics that are specific to particular types of VNF are more appropriate for service specific diagnostics.

2. Real-time Analytics and Orchestration Use Cases

A real-time analytics application periodically collects metrics (also called information in this document) from individual VMs, VNFs, physical servers, network elements etc. regarding various sub-systems such as compute (e.g. dynamic CPU utilization), storage (e.g. dynamic capacity usage), network (e.g. dynamic bandwidth utilization), energy (e.g. dynamic power consumption) through polling. The real-time analytics application computes the average utilization for VMs, VNFs, physical servers, networks etc. regarding the various sub-systems such as compute (e.g. average CPU utilization), storage (e.g. average capacity usage), network (e.g. average bandwidth utilization), energy (e.g. average power consumption).

Using the average utilization information, the real-time analytics application provides real-time visibility into the operating point of the VNF in the NFV Node thus driving operational efficiency.

The NFV orchestrator uses the average utilization information from the real-time analytics application to determine the appropriate time to scale up/down the running software instances. Typically the thresholds for scale up/down are manually programmed into the system - this may not be performance optimal since the workloads and deployment scenarios can substantially vary.

In addition, predictive analytics based on machine learning techniques [MACHINE-LEARNING-BOOK] can be used by the real-time analytics application to automatically determine the appropriate thresholds for scale up/down the running software instances for differing workloads including events related to social behavior (think of a YouTube video going viral) and deployment scenarios. This information can be used by the orchestrator for optimizing overall performance and maximizing energy efficiency. Maximizing energy efficiency comes from the fact that by determining the appropriate thresholds for scale up/down the workloads can be consolidated into a minimum set of physical resources so the rest of the unused physical resources can be completely powered off to avoid

any idle power consumption. [SPEC-BENCHMARK] analyzes the power profile of physical servers from various vendors; the active idle power consumption of physical servers could be as much as 30%.

2.1. Enhancements to Real-time Analytics Application

2.1.1. Distributed Predictive Analytics

A real-time analytics application could be notified of significant events by individual running software instances of VMs, VNFs etc. or by infrastructure elements such as physical servers, hypervisors etc. This helps reduce the rate of polling by the real-time analytics application and also helps in reacting to significant events such as overload much faster. The challenge in this case is to determine the appropriate thresholds (e.g. average power consumption has been higher than x Watts for t seconds) for event notification.

Predictive analytics engines which use machine learning techniques [MACHINE-LEARNING-BOOK] can be used to determine the appropriate thresholds per running software instance and infrastructure element for different workloads and deployment scenarios. These predictive analytics engines can run in various nodes in the infrastructure in a distributed predictive analytics architectural framework.

2.1.2. Detecting Noisy Neighbors

In the context of multiple VNFs, "Noisy Neighbor Effect" could be defined as follows: the VM running for one VNF can affect the performance of a VM running for another VNF in the case where they are using the same physical resources (physical servers, physical network elements). A real-time analytics application could help in detecting and mitigating the noisy neighbor effect. A good example is the case where the VMs running for two VNFs share the same physical server, are memory access intensive (load balancers, firewalls etc.) and have correlated memory access patterns for the given workload and deployment scenario.

Real-time big data analytics techniques [RT-ANALYTICS-BOOK] can be used by the analytics application to determine such correlation patterns which can affect performance in real-time. Additionally, predictive analytics based on machine learning techniques [MACHINE-LEARNING-BOOK] can be used to predict the frequency and duration of such correlation patterns. This information can be used to create dynamic anti-affinity rules for VM placement and migration including redundancy considerations - e.g. VMs of VNF "A" cannot co-exist with VMs of VNF "B".

2.1.3. Addressing security issues due to inconsistent configuration

NFV configuration is expected to be dynamic, especially in the edge NFV PoPs where capacity is limited; a very good example is handling a viral event such as mobile gaming application. While autonomic networking techniques could be used to automate the configuration process including modular updates, it is important to take into account that incomplete and/or inconsistent configuration may lead to security issues. Distributed VNF implementations (e.g. VMs of single VNF which span different physical servers) typically use an eventually consistent configuration model [CAP-THEOREM] for scalability reasons -- this poses additional security challenges.

Real-time analytics techniques [RT-ANALYTICS-BOOK] can be used by the analytics application to determine communication pattern anomalies due to incomplete and/or inconsistent configuration in real-time by analyzing event logs. Additionally, predictive analytics based on machine learning techniques [MACHINE-LEARNING-BOOK] can be used to predict the frequency and duration of such communication pattern anomalies. A simple example is a flow-specific firewall rule which never got installed due to reasons such as control plane messaging issues, data plane table full condition etc.

3. Real-time Analytics and Orchestration Architectural Framework

TBD

4. Summary

TBD

5. Future Work

TBD

6. IANA Considerations

This draft does not have any IANA considerations.

7. Security Considerations

8. Contributors

9. Acknowledgements

None.

10. References

10.1. Normative References

10.2. Informative References

[ETSI-NFV-WHITE] "ETSI NFV White Paper,"
http://portal.etsi.org/NFV/NFV_White_Paper.pdf

[ETSI-NFV-USE-CASES] "ETSI NFV Use Cases,"
http://www.etsi.org/deliver/etsi_gs/NFV/001_099/001/01.01.01_60/gs_NFV001v010101p.pdf

[ETSI-NFV-REQ] "ETSI NFV Virtualization Requirements,"
http://www.etsi.org/deliver/etsi_gs/NFV/001_099/004/01.01.01_60/gs_NFV004v010101p.pdf

[ETSI-NFV-ARCH] "ETSI NFV Architectural Framework,"
http://www.etsi.org/deliver/etsi_gs/NFV/001_099/002/01.01.01_60/gs_NFV002v010101p.pdf

[ETSI-NFV-TERM] "Terminology for Main Concepts in NFV,"
http://www.etsi.org/deliver/etsi_gs/NFV/001_099/003/01.01.01_60/gs_nfv003v010101p.pdf

[OPENSTACK] "OpenStack Open Source Software,"
https://www.openstack.org/

[OPENSTACK-CONGRESS-POLICY-ENGINE] "A policy as a service open source project in OpenStack,"
https://wiki.openstack.org/wiki/Congress

[OPENSTACK-CELIOMETER-MEASUREMENT] "OpenStack Celiometer,"
http://docs.openstack.org/developer/ceilometer/measurements.html

[OPENSTACK-NOVA-COMPUTE] "OpenStack Nova,"
https://wiki.openstack.org/wiki/Nova

[NFV-MANO-SPEC] "NFV Management and Orchestration Framework Specification,"
http://docbox.etsi.org/ISG/NFV/Open/Latest_Drafts/NFV-MAN001v061-%20management%20and%20orchestration.pdf

[BIN-PACK] Coffman, Jr., E., M. Garey, and D. Johnson. Approximation Algorithms for Bin-Packing -- An Updated Survey. In Algorithm Design for Computer System Design, ed. by Ausiello, Lucertini, and Serafini. Springer-Verlag, 1984.

[SPEC-BENCHMARK] "SPEC Benchmark Results: HP Proliant DL380p Rack Server," http://i.dell.com/sites/doccontent/shared-content/data-sheets/en/Documents/Comparing-Dell-R720-and-HP-Proliant-DL380p-Gen8-Servers.pdf

[CAP-THEOREM] Eric Brewer, "CAP twelve years later: How the "rules" have changed", IEEE Explore, Volume 45, Issue 2 (2012), pg. 23-29.

[MACHINE-LEARNING-BOOK] Ian H. Witten et al., "Practical Machine Learning Tools and Techniques, Third Edition," Morgan Kaufmann, 2011

[RT-ANALYTICS-BOOK] Byron Ellis, "Real-Time Analytics: Techniques to Analyze and Visualize Streaming Data," Wiley, 2014

Authors' Addresses

Ram (Ramki) Krishnan
Brocade Communications
ramk@brocade.com

Dilip Krishnaswamy
IBM Research
dilikris@in.ibm.com

Diego Lopez
Telefonica I+D
Don Ramon de la Cruz, 82
Madrid, 28006, Spain
+34 913 129 041
diego.r.lopez@telefonica.com

Asif Qamar
Evolv
asif@asifqamar.com

Internet Research Task Force (IRTF)
Internet-Draft
Intended status: Informational
Expires: September 10, 2015

S. Lee
ETRI
S. Pack
KU
M-K. Shin
ETRI
E. Paik
KT

March 9, 2015

Resource Management in Service Chaining
draft-lee-nfvrg-resource-management-service-chain-01

Abstract

This document specifies problem definition and use cases of NFV resource management in service chaining for path optimization, traffic optimization, failover, load balancing, etc. It further describes design considerations and relevant framework for the resource management capability that dynamically creates and updates network forwarding paths (NFPs) considering resource constraints of NFV infrastructure.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 10, 2015.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents

(http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction

Network Functions Virtualisation (NFV) [ETSI-NFV-WHITE] offers a new way to design, deploy and manage network services. The network service can be composed of one or more network functions and NFV relocates the network functions from dedicated hardware appliances to generic servers, so they can run in software. Using these virtualized network functions (VNFs), the network service can be described by a service chain (or VNF forwarding graph; VNF-FG) which defines an ordered sequence of VNFs for the composed service. The VNF-FG can be instantiated by creating or selecting VNF instances and virtual links (VLs) among them, which results in a network forwarding path (NFP).

The performance or state of the NFP depends on the ones of underlying NFVI resources including VNF instances (VNF-Is) and VLs. For example, if one of the VNF instances in a NFP gets failed, the whole network service using the NFP also gets failed. Thus, the VNF instances per NFP need to be carefully selected at VNF-FG instantiation or dynamically replaced by other VNF instances at run-time for better performance and resilience of the NFP.

The resource placement problem in service chains matters not only to the quality of NFPs but also to the optimized use of NFVI resources. For example, if some of the VNF instances and VLs are selected to constitute a NFP but the others are not, the processing and bandwidth burden will converge on those VNF instances and VLs, which results in scalability problem.

This document addresses resource management problem in service chaining to optimize the NFP quality and resource usage. It provides the relevant use cases of the resource management such as traffic optimization, failover, load balancing and further describes design considerations and relevant framework for the resource management capability that dynamically creates and updates NFPs considering resource state of VNF instances.

This document mainly focuses on the resource capability in the ETSI NFV framework [ETSI-NFV-ARCH] but also studies its applicability to the control plane of SFC architecture [I-D.ietf-sfc-architecture].

2. Terminology

This document uses the following terms and most of them were reproduced from [ETSI-NFV-TERM].

- Network Functions (NF): A functional building block within a network infrastructure, which has well-defined external interfaces and a well-defined functional behavior.
- Network service: A composition of network functions and defined by its functional and behavioural specification.
- NFV Framework: The totality of all entities, reference points, information models and other constructs defined by the specifications published by the ETSI ISG NFV.
- Virtualised Network Function (VNF): An implementation of an NF that can be deployed on a Network Function Virtualisation Infrastructure (NFVI).
- NFV Infrastructure (NFVI): The NFV-Infrastructure is the totality of all hardware and software components which build up the environment in which VNFs are deployed.
- NF Forwarding Graph: A graph of logical links connecting NF nodes for the purpose of describing traffic flow between these network functions.

- o VNF Forwarding Graph (VNF-FG): A NF forwarding graph where at least one node is a VNF.

- o Virtual Link: A set of connection points along with the connectivity relationship between them and any associated target performance metrics (e.g. bandwidth, latency, QoS). The Virtual Link can interconnect two or more entities (VNF components, VNFs, or PNFs).

- o Scaling: Ability to dynamically extend/reduce resources granted to the Virtual Network Function (VNF) as needed.

3. Resource management in service chain

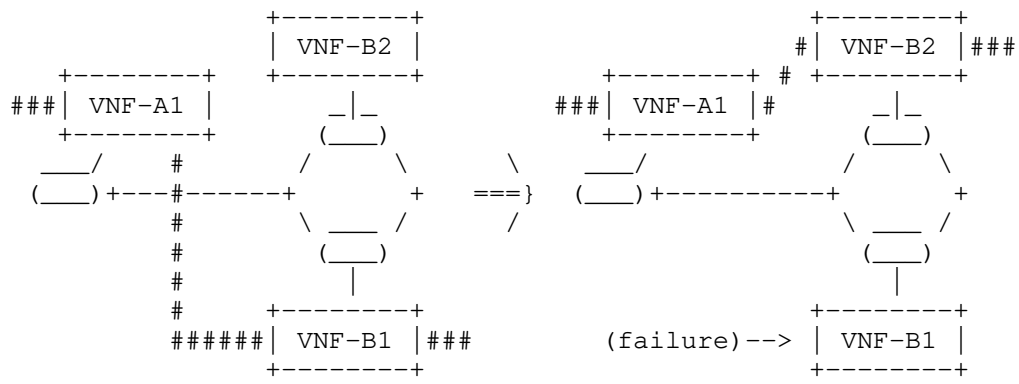
The goal of the resource management is to optimize the quality of network services and resource usage of NFVI. To meet this goal, NFPs of the network services need to consider the state of NFV resources (such as VNF instances or virtual links) at construction. The NFPs also need to dynamically adapt to the changes of the resource state at run-time, such as availability, load, and topological locations of VNF instances; latency and bandwidth of virtual links. The adaptation of NFPs can be executed by monitoring the resource state of VNF instances and VLs and replacing the original VNF instances of the NFP with new VNF instances that constitute a NFP with better performance. This functionality can be a part of Orchestrator functional building block in the NFV framework [ETSI-NFV-MANO] but it needs further study.

3.1. Use cases

In this section, several (but not exhausted) use cases for resource management in service chaining are provided: fail-over, path optimization, traffic optimization, load balancing, and energy efficiency.

Fail-over

When one of VNF instances in a NFP gets failed to run due to failure of its VM or underlying network, the whole chain of network service also gets failed. For service continuity, the failure of VNF instance needs to be detected and the failed one needs to be replaced with the other one which is available to use. Figure 1 presents an example of the fail-over use case. A network service is defined as a chain of VNF-A and VNF-B; and the service chain is instantiated with VNF-A1 and VNF-B1 which are instances of VNF-A and VNF-B respectively. In the meantime, failure of VNF-B1 is detected so that VNF-B2 replaces the failed one for fail-over of the NFP.



NFP

Figure 1: A fail-over use case

Path optimization

Traffic for a network service traverses all of the VNF instances and the connecting VLs given by a NFP to reach a target end point. Thus, quality of the network service depends on the the resource constraints (e.g., processing power, bandwidth, topological locations, latency) of VNF instances and VLs. In order to optimize the path of the network service, the resource constraints of VNF instances and VLs need to be considered at constructing NFPs. Since the resource state may vary in time during the service, NFPs also need to adapt to the changes of resource constraints of the VNF instances and VLs by monitoring and replacing them at run-time.

Traffic optimization

A network operator may provide multiple network services with different VNF-FGs and different flows of traffic traverse between source and destination end-points along the VNF-FGs. For efficiency of network management resource usage, the NFPs need to be built as to localize the traffic flows or as to avoid bottleneck links shared by multiple traffic flows. In this case, multiple NFV instances of different NFPs need to be considered together at constructing a new NFP or adapting one.

Load balancing

A single VNF instances may be shared by multiple traffic flows of the same of different network services. In order to avoid bottleneck

points due to overloaded NFV instances, NFPs need to be constructed or maintained to distribute workloads of the shared VNF instances.

Energy efficiency

Energy efficiency in the network is getting important to reduce impact on the environment so that energy consumption of VNF instances using VNFI resources (e.g., compute, storage, I/O) needs to be considered at NFP construction or adaptation. For example, a NFP can be constructed as to make traffic flows aggregated into a limited number of VNF instances as much as its performance is preserved in a certain level.

3.2. Design considerations

To support the aforementioned use cases, it is required to support resource management capability which provides service chain (or NFP) construction and adaptation by considering resource state or constraints of VNF instances and virtual links among them. The resource management operations for service chain construction and adaptation can be divided into several sub-actions:

- Select a VNF instance
- Evaluate a VNF instance and a virtual link
- Replace a VNF instance to update a NFP
- Monitor state or resource constraints of a VNF instance and a virtual link
- Migrate a VNF instance to another ones in different locations

Note: While scaling-in/out or -up/down of VNF instances is one of the essential actions for NFV resource management, it is a different approach with a finer granularity than service chain adaptation. The scaling approach may be integrated together with the service chain adaptaiton but it is still under study.

As listed above, VNF instances are selected or replaced according to monitoring or evaluation results of performance metrics of the VNF instances and virtual links. Studies about evaluation methodologies and performance metrics for VNF instances and NFVI resources can be found at [ETSI-NFV-PER001] [I-D.liu-bmwg-virtual-network-benchmark] [I-D.morton-bmwg-virtual-net]. The performance metrics of VNF instances and virtual links specific to service chain construction and adaptation can be defined as follows:

- o availability (or failure) of a VNF instance and a virtual link
- o a topological location of a VNF instance
- o a utilization rate of a VNF instance
- o a throughput of a VNF instance
- o energy consumption of VNF instance
- o bandwidth of a virtual link
- o latency of a virtual link

3.3. Framework

The resource management functionality for dynamic service chain adaptation takes role of NFV orchestration with support of VNF manager and Virtualised Infrastructure Manager (VIM) in the NFV framework [ETSI-NFV-ARCH]. Detailed functional building block and interfaces are still under study.

4. Applicability to SFC

4.1. Related works in IETF SFC WG

IETF SFC WG provides a new service deployment model that delivers the traffic along the predefined logical paths of service functions (SFs), called service function chains (SFCs) with no regard of network topologies or transport mechanisms. Basic concept of the service function chaining is similar to VNF-FG where a network service is composed of SFs and deployed by making traffic flows traversed instances of the SFs in a pre-defined order.

There are several works in progress in IETF SFC WG for resource management of service chaining. [I-D.ietf-sfc-architecture] defines SFC control plane that selects specific SFs for a requested SFC, either statically or dynamically but details are currently outside the scope of the document. There are other works [I-D.ww-sfc-control-plane] [I-D.lee-sfc-dynamic-instantiation] [I-D.krishnan-sfc-oam-req-framework] [I-D.aldrin-sfc-oam-framework] which define the control plane functionality for service function chain construction and adaptation but details are still under study. While [I-D.dunbar-sfc-fun-instances-restoration] and [I-D.meng-sfc-chain-redundancy] provide detailed mechanisms of service chain adaptation, they focus only on resilience or fail-over of service function chains.

4.2. Integration in SFC control-plane architecture

In SFC WG, [I-D.ww-sfc-control-plane] defines a generic architecture of SFC control plane with well-defined functional building blocks and interfaces as follows:

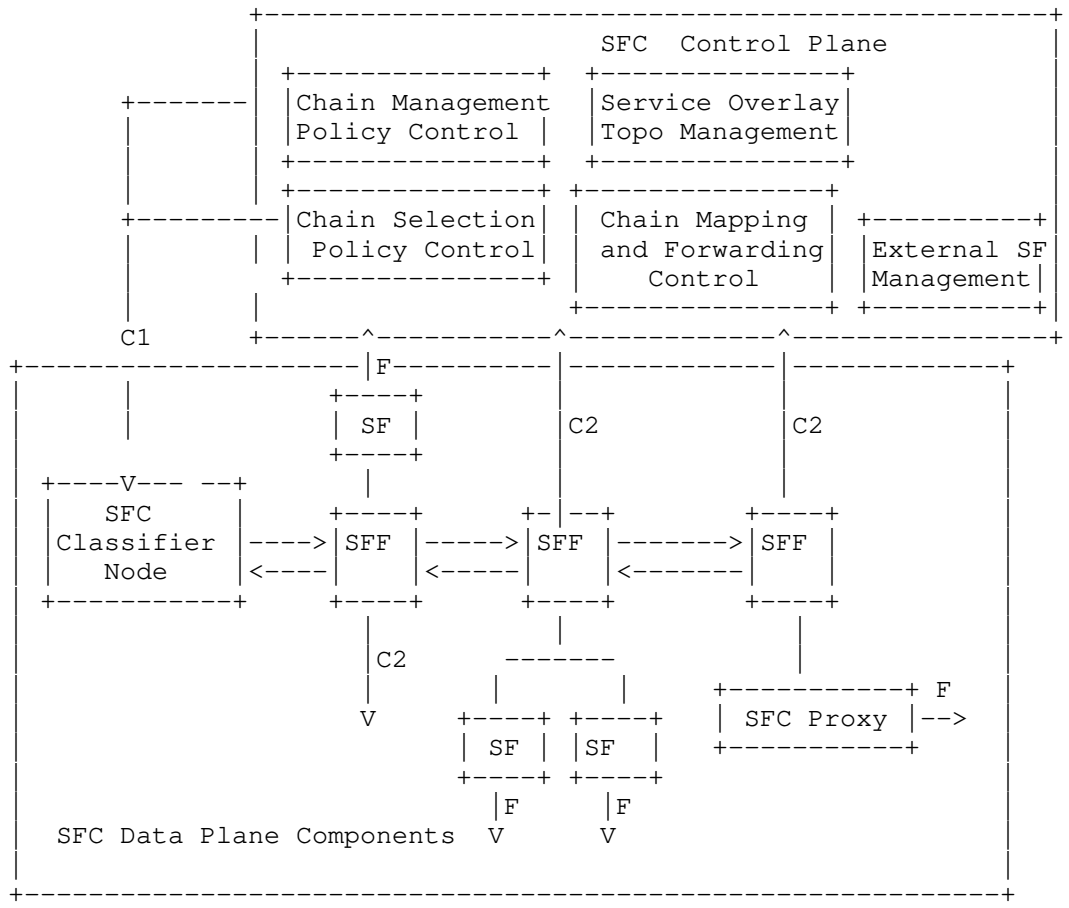


Figure 2: SFC control plane architecture

The service chain adaptation addressed in this document may be integrated into the Chain Mapping and Forwarding Control functional block and may use the C2 and F interfaces for monitoring or collecting the resource constraints of VNF intances and VLs.

Note that SFC does not assume that Service Functions are virtualized. Thus, the parameters of resource constraints may differ, and it needs further study for integration.

5. Security Considerations

TBD.

6. IANA Considerations

TBD.

7. References

7.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

7.2. Informative References

[ETSI-NFV-ARCH]
ETSI, "ETSI NFV Architectural Framework v1.1.1", October 2013.

[ETSI-NFV-MANO]
ETSI, "Network Function Virtualization (NFV) Management and Orchestration V0.6.3", October 2014.

[ETSI-NFV-PER001]
ETSI, "Network Function Virtualization: Performance and Portability Best Practices v1.1.1", June 2014.

[ETSI-NFV-TERM]
ETSI, "NFV Terminology for Main Concepts in NFV", October 2013.

[ETSI-NFV-WHITE]
ETSI, "NFV Whitepaper 2", October 2013.

[I-D.aldrin-sfc-oam-framework]
Aldrin, S., Krishnan, R., Akiya, N., Pignataro, C., and A. Ghanwani, "Service Function Chaining Operation, Administration and Maintenance Framework", draft-aldrin-sfc-oam-framework-01 (work in progress), October 2014.

[I-D.dunbar-sfc-fun-instances-restoration]
Dunbar, L. and A. Malis, "Framework for Service Function Instances Restoration", draft-dunbar-sfc-fun-instances-restoration-00 (work in progress), April 2014.

[I-D.ietf-sfc-architecture]
Halpern, J. and C. Pignataro, "Service Function Chaining (SFC) Architecture", draft-ietf-sfc-architecture-07 (work in progress), March 2015.

[I-D.krishnan-sfc-oam-req-framework]
Krishnan, R., Ghanwani, A., Gutierrez, P., Lopez, D., Halpern, J., Kini, S., and A. Reid, "SFC OAM Requirements and Framework", draft-krishnan-sfc-oam-req-framework-00 (work in progress), July 2014.

[I-D.lee-sfc-dynamic-instantiation]
Lee, S., Pack, S., Shin, M., and E. Paik, "SFC dynamic instantiation", draft-lee-sfc-dynamic-instantiation-01 (work in progress), October 2014.

[I-D.liu-bmwg-virtual-network-benchmark]
Liu, V., Liu, D., Mandeville, B., Hickman, B., and G. Zhang, "Benchmarking Methodology for Virtualization Network Performance", draft-liu-bmwg-virtual-network-benchmark-00 (work in progress), July 2014.

[I-D.meng-sfc-chain-redundancy]
Wu, C., Meng, W., and C. Wang, "Redundancy Mechanism for Service Function Chains", draft-meng-sfc-chain-redundancy-01 (work in progress), December 2014.

[I-D.morton-bmwg-virtual-net]
Morton, A., "Considerations for Benchmarking Virtual Network Functions and Their Infrastructure", draft-morton-bmwg-virtual-net-03 (work in progress), February 2015.

[I-D.ww-sfc-control-plane]
Li, H., Wu, Q., Boucadair, M., Jacquenet, C., Haeffner, W., Lee, S., and R. Parker, "Service Function Chaining (SFC) Control Plane Achitecture", draft-ww-sfc-control-plane-04 (work in progress), March 2015.

Authors' Addresses

Seungik Lee
ETRI
218 Gajeong-ro Yuseung-Gu
Daejeon 305-700
Korea

Phone: +82 42 860 1483
Email: seungiklee@etri.re.kr

Sangheon Pack
Korea University
145 Anam-ro, Seongbuk-gu
Seoul 136-701
Korea

Phone: +82 2 3290 4825
Email: shpack@korea.ac.kr

Myung-Ki Shin
ETRI
218 Gajeong-ro Yuseung-Gu
Daejeon 305-700
Korea

Phone: +82 42 860 4847
Email: mkshin@etri.re.kr

EunKyoung Paik
KT
17 Woomyeon-dong, Seocho-gu
Seoul 137-792
Korea

Phone: +82 2 526 5233
Email: eun.paik@kt.com

NFVRG
Internet-Draft
Intended status: Informational
Expires: May 5, 2016

M-K. Shin
ETRI
K. Nam
Friesty
S. Pack
KU
S. Lee
ETRI
R. Krishnan
Dell
T. Kim
LG U+

October 5, 2015

Verification of NFV Services : Problem Statement and Challenges
draft-shin-nfvrg-service-verification-04

Abstract

NFV relocates network functions from dedicated hardware appliances to generic servers, so they can run in software. However, incomplete or inconsistent configuration of virtualized network functions (VNF) and forwarding graph (FG, aka service chain) could cause break-down of the supporting infrastructure. In this sense, verification is critical for network operators to check their requirements and network properties are correctly enforced in the supporting infrastructures. Recognizing these problems, we discuss key properties to be checked on NFV-enabled services. Also, we present challenging issues related to verification in NFV environments.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on May 2, 2016.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction

NFV is a network architecture concept that proposes using IT virtualization related technologies, to virtualize entire classes of network service functions into building blocks that may be connected, or chained, together to create network services. NFV service is defined as a composition of network functions and described by its functional and behavioral specification, where network functions (i.e., firewall, DPI, SSL, load balancer, NAT, AAA, etc.) are well-defined, hence both their functional behavior as well as their external interfaces are described in each specifications.

In NFV, a VNF is a software package that implements such network functions. A VNF can be decomposed into smaller functional modules or APIs for scalability, reusability, and/or faster response [ETSI-NFV-Arch],[ETSI-NFV-MANO]. This modular updates or composition for a network function may lead to many other verification or security issues. In addition, a set of ordered network functions which build FGs may be connected, or chained, together to create an end-to-end network service. Multiple of VNFs can be composed together to reduce management and VNF FGs. While autonomic networking techniques could be used to automate the configuration process including FG updates, it is important to take into account that incomplete and/or inconsistent configuration may lead to verification issues. Moreover, automation of NFV process with integration of SDN may lead the network services to be more error-prone. In this sense, we need to identify and verify key properties to be correct before VNFs and FGs are physically placed and realized in the supporting infrastructure.

1.1. Terminology

This document draws freely on the terminology defined in [ETSI-NFV-Arch].

2. Problem statement

The verification services should be able to check the following properties:

2.1. Dependencies of network service components in NFV framework

In NFV framework, there exist several network service components including NFVI, VNFs, MANO, etc. as well as network controller and switches to realize end-to-end network services. Unfortunately, these components have intricate dependencies that make operation incorrect. In this case, there is inconsistency between states stored and managed in VNF FGs and network tables (e.g., flow tables), due to

communication delays and/or configuration errors. For example, if a VNF is replicated into the other same one for the purpose of load balance and a new FG is established through the copied one, but all the state/DBs replication is not finished yet due to delays, this can be lead to unexpected behaviors or errors of the network service. Therefore, these dependencies make it difficult to correctly compose NFV-enabled end-to-end network services.

2.2. Invariant and error check in VNF FGs

In VNF FGs, an infinite loop construction should be avoided and verified. Let us consider the example. Two VNF A and VNF B locate in the same service node X whereas another VNF C resides in other service node Y [SIGCOMM-Gember]. Also, the flow direction is from X toY, and the given forwarding rule is A->C->B. In such a case, service node Y can receive two ambiguous flows from VNF A: 1) one flow processed by VNF A and 2) another flow processed by VNF A, B, and C. For the former case, the flow should be processed by VNF C whereas the latter flow should be further routed to next service nodes. If these two flows cannot be distinguished, service node Y can forward the flow to service node X even for the latter case and a loop can be formed. To avoid the infinite loop formation, the forwarding path over VNF FG should be checked in advance with the consideration of physical placement of VNF among service nodes. Also, reactive verification may be necessary, since infinite loop formation may not be preventable in cases where configuration change is happening with live traffic.

In addition, isolation between VNFs (e.g. confliction of properties or interference between VNFs) and consistent ordering of VNF FGs should be always checked and maintained.

2.3. Load balancing among VNF instances

In VNF FG, different number of VNF instances can be activated on several service nodes to carry out the given task. In such a situation, load balancing among the VNF instances is one of the most important considerations. In particular, the status in resource usage of each service node can be different and thus appropriate amount of jobs should be distributed to the VNF instances. To guarantee well-balanced load among VNF instances, the correctness of hash functions for load balancing needs to be verified. Moreover, when VNF instances locate in physically different service nodes, simple verification of load balancing in terms of resource usage is not sufficient because different service nodes experience diverse network conditions (e.g., different levels of network congestion)[ONS- Gember]. Therefore, it is needed to monitor global network condition as well as local resource condition to achieve the network-wide load balancing in VNF

FGs. Also, whether the monitoring function for network/compute/storage resources is correctly working should be checked.

2.4. Policy and state consistency on NFV services

In VNF FG, policy to specific users can be dynamically changed. For example, a DPI VNF can be applied only in the daytime in order to prohibit from watching adult contents while no DPI VNFs applied during the nighttime. When the policy is changed, the changed policy should be reconfigured in VNF service nodes as soon as possible. If the reconfiguration procedure is delayed, inconsistent policies may exist in service nodes. Consequently, policy inconsistency or confliction needs to be checked. Also in some situations, states for VNF instances may be conflicted or inconsistent. Especially when a new VNF instance is instantiated for scale-up and multiple VNF instances are running, these multiple VNF instances may have inconsistent states owing to inappropriate instantiation procedure [SIGCOMM-Gember]. In particular, since the internal states of VNF instances (e.g., the instantaneous state of CPU, register, and memory in virtual machine) are not easily-visible, a new way to check the VNF internal states should be devised.

2.5. Performance

In VNF FG, VNF instances can locate in different service nodes and these service nodes have different load status and network conditions. Consequently, the overall throughput of VNF FG is severely affected by the service nodes running VNF instances. For example, if a VNF instance locates in a heavily loaded service node, the service time at the service node will be increased. In addition, when a VNF FG includes a bottleneck link with network congestion, the end-to-end performance (e.g., latency and throughput) in the VNF FG can be degraded. Therefore, the identification of bottleneck link and node is the first step for performance verification or guarantee of the VNF FG [ONS-Gember]. After detecting the bottleneck link/node, the VNF requiring scale up or down can be identified and the relocation of VNF instance among service nodes can be determined

2.6. Security

How to verify security holes in VNF FG is another important consideration. In terms of security services, authentication, data integrity, confidentiality, and replay protection should be provided. On the other hand, several VNFs (e.g., NAT) can modify or update packet headers and payload. In these environments, it is difficult to protect the integrity of flows traversing such VNFs. Another security concern in the VNF FG is distributed denial of service (DDoS) to a

specific service node. If an attacker floods packets to a target service node, the target service node cannot perform its functions correctly. Therefore, such security attacks in the VNF FG should be detected and handled in an efficient manner. In the case of DDoS, adding a DDoS appliance as the first element in the service chain would help alleviate the problem. Moreover, unknown or unauthorized VNFs can run and thus how to identify those problems is another security challenge.

3. Examples - NS policy conflict with NFVI policy

Another target of NFV verification is conflict of NS policies against global network policy, called NFVI policy.

NFV allocates and manages NFVI resources for a network service according to an NS policy given in the network service descriptor (NSD), which describes how to govern NFVI resources for VNF instances and VL instances to support KPIs of the network service. Example factors of the NS policy are resource constraints (or deployment flavor), affinity/anti-affinity, scaling, fault and performance management, NS topology, etc.

For a network-wide (or NS-wide) management of NFVI, NFVI policy (or global network policy) can be provided to describe how to govern the NFVI resources for optimized use of the infrastructure resources (e.g., energy efficiency and load balancing) rather than optimized performance of a single network service. Example factors of the NFVI policy are NFVI resource access control, reservation and/or allocation policies, placement optimization based on affinity and/or anti-affinity rules, geography and/or regulatory rules, resource usage, etc.

While both of the policies define the requirements for resource allocation, scheduling, and management, the NS policy is about a single network service; and the NFVI policy is about the shared NFVI resources, which may affect all of the given network services globally. Thus, some of NS and NFVI policies may be inconsistency with each other when they have contradictive resource constraints on the shared NFVI resources. Examples of the policy conflicts are as follows:

<Example conflict case #1>

- NS policy of NS_A (composed of VNF_A and VNF_B)
  - Resource constraints: 3 CPU core for VNF_A and 2 CPU core for VNF_B
  - Affinity rule between VNF_A and VNF_B

- NFVI policy
  - No more than 4 CPU cores per physical host

- Conflict case
  - The NS policy cannot be met within the NFVI policy

<Example conflict case #2>

- NS policy of NS_B (composed of VNF_A and VNF_B)
  - Affinity rule between VNF_A and VNF_B

- NFVI policy
  - Place VM whose outbound traffic is larger than 100Mbps at POP_A
  - Place VM whose outbound traffic is smaller than 100Mbps at POP_B

- Conflict case
  - If VNF_A and VNF_B generate traffic in 150Mbps and 50Mbps, respectively,
  - VNF_A and VNF_B need to be placed at POP_A and POP_B, respectively according to the NFVI policy
  - But it will violate the affinity rule given in the NS policy

<Example conflict case #3>

- NS policy of NS_C (composed of VNF_A and VNF_B)
  - Resource constraints: VNF_A and VNF_B exist in the same POP
  - Auto-scaling policy: if VNF_A has more than 300K CPS, scale-out

- NFVI policy
  - No more than 10 VMs per physical host in POP_A

- Conflict case
  - If CPS of VNF_A in POP_A gets more than 300K CPS,
  - and if there is no such physical host in the POP_A whose VMs are smaller than 10,
  - VNF_A need to be scaled-out to other POP than POP_A according to the NFVI policy
  - But it will violate the NS policy

4. Requirements of verification framework

A verification framework for NFV-based services needs to satisfy the following requirements:.

- R1 : It should be able to check global and local properties and invariants. Global properties and invariants relate to the entire VNFs, and local properties and invariants relates to

the specific domain or resources that some of the VNFs are using. For example, Loop-freeness and isolation between VNFs can be regarded as global. The policies that are related only to the specific network controllers or devices are local.

- o R2 : It should be able to access to the entire network states whenever verification tasks are started. It can directly manage the states of network and NFV-based services through databases or any solution that specializes in dealing with the network topology and configurations, or can utilize the functions provided by NFV M&O and VNFI solutions to get or set the states at any time.

- o R3 : It should be independent from specific solutions and frameworks, and provide standard APIs.

- o R4 : It should process standard protocols such as NetConf, YANG, OpenFlow, and northbound and southbound interfaces that are related network configurations, and used by OSS.

5. Challenging issues

There are emerging challenges that the verification services face with.

5.1. Consistency check in distributed state

Basically, NFV states as well as SDN controllers are distributed. writing code that works correctly in a distributed setting is very hard. Therefore, distributed state management and consistency check has challenging issues. Some open source project such as ONOS offers a core set of primitives to manage this complexity. RAFT algorithm [RAFT] is used for distribution and replication. Similarly, Open daylight project has a clustering concept to management distributed state. There is no "one-size-fits-all" solution for control plane data consistency.

5.2. Intent-based service composition

Recently, Intent-based high-level language is newly proposed and discussed in open source project. The Intent allows for a descriptive way to get what is desired from the infrastructure, unlike the current NFV description and SDN interfaces which are based on describing how to provide different services. This Intent will accommodate orchestration services and network and business oriented SDN/NFV applications, including OpenStack Neutron, Service Function

Chaining, and Group Based Policy. A Intent compiler that translates and compiles it into low level instructions (e.g., SDN controller/OpenStack primitives) for network service components. In this sense, error checking and debugging are critical for reliable Intent-based service composition.

5.3. Finding infinite loops

General solutions for the infinite loop can lead to intractable problem (e.g. the halting problem). To make the verification practical and minimize the complexity, some of the restrictions are required. Finding cycle can be processed in polynomial time but the restriction could be too much for some cases that service functions or network flows requires finite loops.

5.4. Live traffic verification

It is known fact that the complexity of verification tasks for the real and big problem is high. A few invariants can be checked in real-time but it would be impossible if the size of VNFs increases or properties to be checked are complex.

5.5. Languages and their semantics

For the verification, configurations and states of VNFs need to be precisely expressed using formal semantics. There are many languages and models, and it is impractical for the verification frameworks to support all of the existing languages and models. Languages and semantic models optimized to the verification framework need to selected or newly developed.

6. Gap analysis - open source projects

Recently, the Open Platform for NFV (OPNFV) community is collaborating on a carrier-grade, integrated, open source platform to accelerate the introduction of new NFV products and services [OPNFV]. Open Daylight (ODL) is also being tightly coupled with this OPNFV platform to integrate SDN controller into NFV framework [ODL].

This clause analyzes the existing open source projects including OPNFV and ODL related to verification of NFV services.

6.1. OPNFV

6.1.1. Doctor

The Doctor project provides a NFVI fault management and maintenance

framework on top of the virtualized infrastructure. The key feature is to notify unavailability of virtualized resources and to recover unavailable VNFs.

While the Doctor project focuses only on faults in NFVI including compute, network, and storage resources, the document discusses broader fault management issues such as break-down of the supporting infrastructure due to incomplete or inconsistent configuration of NFV services.

6.1.2. Prediction

The Prediction project provides a data collection for failure prediction framework. The failure prediction framework diagnoses or verifies which entity is suspected to be progressing towards a failure and which VNFs might be affected due to the predicted anomaly.

While the Prediction project focuses only on fault prediction in NFVI compute, network, and storage resources, the document includes broader fault management and prediction issues such as faults in the NFV service deployment and operation.

6.1.3. Resource Scheduler

The Resource Scheduler project provides an enhanced scheduler for optimizing the performance of the VNFs. In particular, this project supports resource isolation. For example, when a VNF strictly requires low latency, strongly isolated compute resources can be allocated to the VNF.

The Resource Scheduler project only focuses on optimizing the performance of individual VNFs without considering the end-to-end performance (e.g., latency and throughput) in NFV services.

6.1.4. Moon

The Moon project implements a security management system for the cloud computing infrastructure. The project also enforces the security managers through various mechanisms, e.g., authorization for access control, firewall for networking, isolation for storage, and logging for tractability.

Note that the main interest of the Moon project is the DDoS attack to a service node and the IDS management for VNFs. A wider range of security issues in the NFV service verification need to be discussed.

6.1.5 Bottlenecks

The Bottlenecks project aims to find system bottlenecks by testing and verifying OPNFV infrastructure in a staging environment before committing it to a production environment. Instead of debugging the deployment in production environment, an automatic method for executing benchmarks to validate the deployment during staging is adopted. For example, the system measures the performance of each VNF by generating workload on VNFs.

The Bottlenecks project does not consider incomplete or inconsistent configurations on NFV services that might cause the system bottlenecks. Furthermore, the Bottlenecks project aims to find system bottlenecks before committing it to a production environment. Meanwhile, the draft also considers how to find bottlenecks in real time.

6.2. ODL

6.2.1. Network Intent Composition

The Network Intent Composition project enables the controller to manage and direct network services and network resources based on intent for network behaviors and network policies. Intents are described to the controller through a new northbound interface, which provides generalized and abstracted policy semantics. Also, the Network Intent Composition project aims to provide advanced composition logic for identifying and resolving intent conflicts across the network applications.

When the reconfiguration upon the policy (i.e, intent) is delayed, policy inconsistency in service nodes may occur after the policy is applied to service nodes. While the Network Intent Composition project resolves such intent conflicts only before they are translated into service nodes, this document covers intent conflicts and inconsistency issues in a broader sense.

6.2.2. Controller Shield

The Controller Shield project proposes to create a repository called unified-security plugin (USecPlugin). The unified-security plugin is a general purpose plugin to provide the controller security information to northbound applications. The security information could be for various purposes such as collating source of different attacks reported in southbound plugins and suspected controller intrusions. Information collected at this plugin can also be used to configure firewalls and create IP blacklists for the network.

In terms of security services, the document covers authentication, data integrity, confidentiality, and replay protection. However, the Controller Shield project only covers authentication, data integrity, and replay protection services where the confidentiality service is not considered.

6.2.3. Defense4All

The Defense4All project proposes a SDN application for detecting and mitigating DDoS attacks. The application communicates with ODL controller via the northbound interface and performs the two main tasks; 1) Monitoring behavior of protected traffic and 2) Diverting attacked traffic to selected attack mitigation systems (AMSs).

While the Defense4All project only focuses on defense system at the controller, this document includes broader defense issues at the service node as well as the controller.

6.3. Summary

The verification functions should spreads over the platforms to accomplish the requirements mentioned in clause 3. The correctness of NFV- based services and their network configurations can be checked in the NFV MANO layer which has the entire states of the VNFs. Each NFVI needs to provide verification layer which composed of policy manager, network database and interfaces (e.g. REST APIs). Local properties and invariants can be verified inside the specific NFVI, and the global properties and invariants can be checked by merging local verification results from the related NFVIs.

The verification service provides verification functions to NFV MANO, NFVI, and any other low-level modules such as SDN controllers. For the platform independency, it provides standard APIs to process the verification tasks. It also uses standard APIs provided by OSS such as OpenStack (Neutron) and Open Daylight. The compiler and interpreter translate standard description languages and protocols into the internal model which optimized to the verification tasks. It can process user-defined properties to be checked as well. The properties to be checked whether they are user-defined or pre-defined invariants are managed by property library. The verifier maintains a set of verification algorithms to check the properties. The network database inside the verification service manages the global network states directly or indirectly.

A PoC can be implemented using OpenStack (Neutron) and Open Daylight. The modules related to verification framework can reside in between network virtualization framework (e.g. OpenStack Neutron) and SDN controller (e.g. Open Daylight). Neutron and Open Daylight uses

standard APIs provided by verification service to accomplish verification tasks. The initial use case for the PoC could be, in particular, any of security, performance, etc as mentioned in clause 2.

7. Security Considerations

As already described in clause 2.6, how to verify security holes in VNF FG is very important consideration. In terms of security services, authentication, data integrity, confidentiality, and replay protection should be provided. On the other hand, potential security concern should be also carefully checked since several VNFs (e.g., NAT) can modify or update packet headers and payload.

8. Acknowledgements

The authors would like to thank formal methods lab members in Korea University for their verification theory support.

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

9.2. Informative References

[ETSI-NFV-Arch] ETSI, "Network Function Virtualisation (NFV); Architectural Framework," 2014.

[ETSI-NFV-MANO] ETSI, "Network Function Virtualiztion (NFV) Management and Orchestration," 2014.

[SIGCOMM-Qazi] Z. Qazi, C. Tu, L. Chiang, R. Miao, V. Sekar, and M. Yu, "SIMPLE-fying Middlebox Policy Enforcement Using SDN," in Proc. ACM SIGCOMM 2013, August 2013.

[ONS-Gember] A. Gember, R. Grandl, A. Anand, T. Benson, and A. Akella, "Stratos: Virtual Middleboxes as First-Class Entities," ONS 2013 and TR.

[SIGCOMM-Gember] A. Gember, R. Viswanathan, C. Prakash, R. Grandl, J. Khalid, S. Das, and A. Akella, "OpenNF: Enabling

Innovation in Network Function Control," in Proc. ACM SIGCOMM 2014, August 2014.

[RAFT] https://raftconsensus.github.io/.

[ODL] "OpenDaylight SDN Controller, "http://www.opendaylight.org/

[OPNFV] "Open Platform for NFV, "https://www.opnfv.org/

Authors' Addresses

Myung-Ki Shin
ETRI
161 Gajeong-dong Yuseng-gu
Daejeon, 305-700
Korea

Phone: +82 42 860 4847
Email: mkshin@etri.re.kr

Ki-Hyuk Nam
Friesty

Email: nam@friesty.com

Sangheon Pack
Korea University

Email: shpack@korea.ac.kr

Seungik Lee
ETRI
161 Gajeong-dong Yuseng-gu
Daejeon, 305-700
Korea

Phone: +82 42 860 1483
Email: seungiklee@etri.re.kr

Ramki Krishnan
Dell

Email: Ramki_Krishnan@dell.com

Tae-wan Kim
LG U+

Phone: +82 10 8080 6603
Email: dm24ks@lguplus.co.kr

NFVRG
Internet-Draft
Intended status: Informational
Expires: January 9, 2017

R. Szabo
A. Csaszar
Ericsson
K. Pentikousis
Travelping
M. Kind
Deutsche Telekom AG
D. Daino
Telecom Italia
Z. Qiang
Ericsson
H. Woesner
BISDN
July 8, 2016

Unifying Carrier and Cloud Networks: Problem Statement and Challenges
draft-unify-nfvrg-challenges-04

Abstract

The introduction of network and service functionality virtualization in carrier-grade networks promises improved operations in terms of flexibility, efficiency, and manageability. In current practice, virtualization is controlled through orchestrator entities that expose programmable interfaces according to the underlying resource types. Typically this means the adoption of, on the one hand, established data center compute/storage and, on the other, network control APIs which were originally developed in isolation. Arguably, the possibility for innovation highly depends on the capabilities and openness of the aforementioned interfaces. This document introduces in simple terms the problems arising when one follows this approach and motivates the need for a high level of programmability beyond policy and service descriptions. This document also summarizes the challenges related to orchestration programming in this unified cloud and carrier network production environment. A subsequent problem is the resource orchestration. This is handled separately in [I-D.caszpe-nfvrg-orchestration-challenges] and will be merged in the next iteration of this document.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute

working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 9, 2017.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

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 speedy service

creation, as discussed in [NSC] and the references therein. For instance, it is not unusual that today an average service creation time cycle exceeds 90 hours, whereas given the recent advances in virtualization and cloudification one would be interested in service creation times in the order of minutes [EU-5GPPP-Contract] if not seconds.

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 [I-D.ietf-sfc-problem-statement]. 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.

Compute virtualization is central to the concept of Network Function Virtualization (NFV). However, carrier-grade services demand that all components of the data path, such as Network Functions (NFs), virtual NFs (VNFs) and virtual links, meet key performance requirements. In this context, the inclusion of Data Center (DC) platforms, such as OpenStack [OpenStack], into the SDN infrastructure is far from trivial.

In this document we examine the problems arising as one combines these two formerly isolated environments in an effort to create a unified production environment and discuss the associated emerging challenges. Our goal is the definition of a production environment that allows multi-vendor and multi-domain operation based on open and interoperable implementations of the key entities described in the remainder of this document.

2. Terms and Definitions

We use the term compute and "compute and storage" interchangeably throughout the document. Moreover, we use the following definitions, as established in [ETSI-NFV-Arch]:

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

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

NFVI: NFV Infrastructure - Collection of NFVI PoPs under one orchestrator.

VNF: Virtualized Network Function - a software-based network function.

VNF FG: Virtualized Network Function Forwarding Graph - an ordered list of VNFs creating a service chain.

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

Further, we make use of the following terms:

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

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

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

LSI: Logical Switch Instance - a software switch instance.

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

UN: Universal Node - an innovative element that integrates and manages in a unified platform both compute and networking components.

3. Motivations

Figure 1 illustrates a simple service graph comprising three network functions (NFs). For the sake of simplicity, we will assume only two types of infrastructure resources, namely SWs and DCs as per the terminology introduced above, and ignore appliance-based NFs for the time being. The goal is to implement the given service based on the available infrastructure resources.

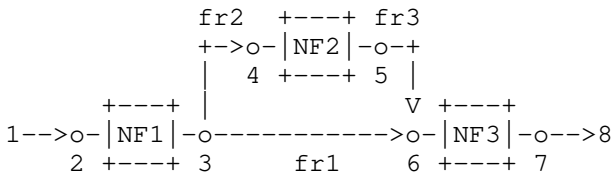


Figure 1: Service graph

The service graph definition contains NF types (NF1, NF2, NF3) along with the

- corresponding ports (NF1:{2,3}; NF2:{4,5}; NF3:{6,7})
- service access points {1,8} corresponding to infrastructure resources,
- definition of forwarding behavior (fr1, fr2, fr3)

The forwarding behavior contains classifications for matching of traffic flows and corresponding outbound forwarding actions.

Assume now that we would like to use the infrastructure (topology, network and software resources) depicted in Figure 2 and Figure 3 to implement the aforementioned service graph. That is, we have three SWs and two Points of Presence (PoPs) with DC software resources at our disposal.

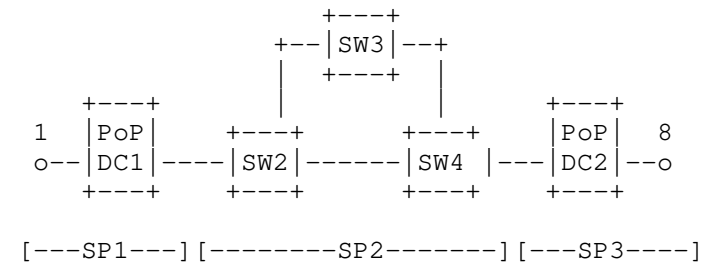


Figure 2: Infrastructure resources

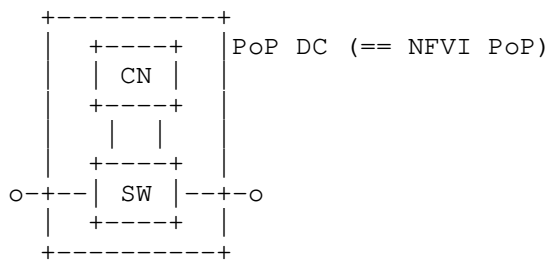


Figure 3: A virtualized Point of Presence (PoP) with software resources (Compute Node - CN)

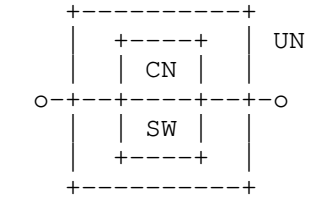


Figure 4: Universal Node - an innovative element that integrates on the same platform both compute and networking components

In the simplest case, all resources would be part of the same service provider (SP) domain. We need to ensure that each entity in Figure 2 can be procured from a different vendor and therefore interoperability is key for multi-vendor NFVI deployment. Without such interoperability different technologies for data center and network operation result in distinct technology domains within a single carrier. Multi-technology barriers start to emerge hindering the full programmability of the NFVI and limiting the potential for rapid service deployment.

We are also interested in a multi-operation environment, where the roles and responsibilities are distributed according to some organizational structure within the organization. Finally, we are interested in multi-provider environment, where different infrastructure resources are available from different service providers (SPs). Figure 2 indicates a multi-provider environment in the lower part of the figure as an example. We expect that this type of deployments will become more common in the future as they are well suited with the elasticity and flexibility requirements [NSC].

Figure 2 also shows the service access points corresponding to the overarching domain view, i.e., {1,8}. In order to deploy the service graph of Figure 1 on the infrastructure resources of Figure 2, we

will need an appropriate mapping which can be implemented in practice.

Figure 3 shows the structure of a PoP DC that presents compute and network resources while Figure 4 shows the structure of the Universal Node (UN), an innovative element that integrates on the same platform both compute and networking components and that could be used in the infrastructure as an alternative to elements depicted in Figure 2 for what concerns network and/or compute resources.

In Figure 5 we illustrate 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.

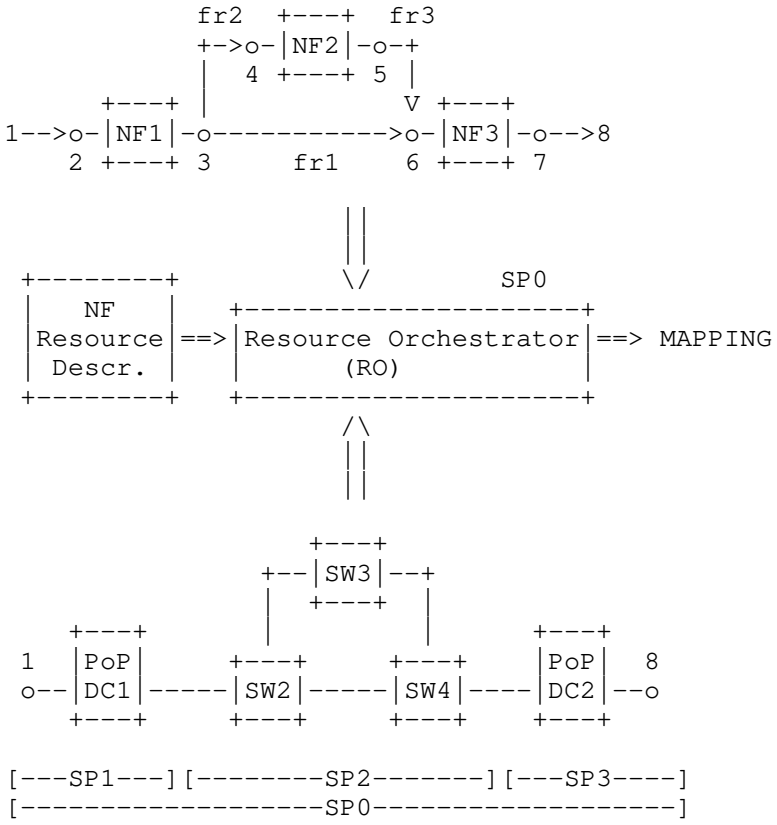


Figure 5: Resource Orchestrator: information base, inputs and output

NF resource descriptions 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 SP1, SP2 and SP3.

In order for the RO to execute the resource mapping (which in general is a hard problem) it needs to operate on the combined control plane illustrated in Figure 6. 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.

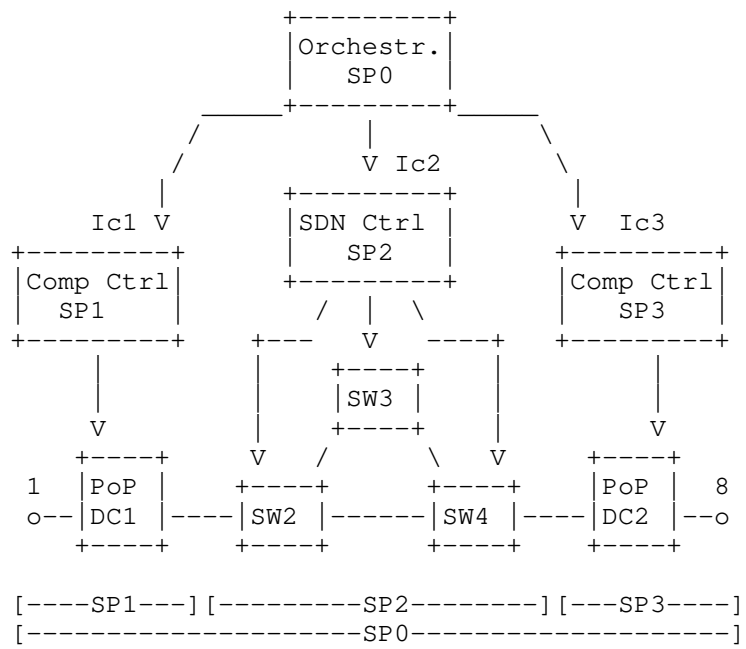


Figure 6: The RO Control Plane view. Control plane interfaces are indicated with (line) arrows. Data plane connections are indicated with simple lines.

In the real-world, however, orchestration operations do not stop, for example, at the DC1 level as depicted in Figure 6. If we (so-to-speak) "zoom into" DC1 we will see a similar pattern and the need to coordinate SW and DC resources within DC1 as illustrated in Figure 7. As depicted, this edge PoP includes compute nodes (CNs) and SWs which in most of the cases will also contain an internal topology.

In Figure 7, IcA is an interface similar to Ic2 in Figure 6, while IcB could be, for example, OpenStack Nova or similar. The Northbound Interface (NBI) to the Compute Controller can use Ic1 or Ic3 as shown in Figure 6.

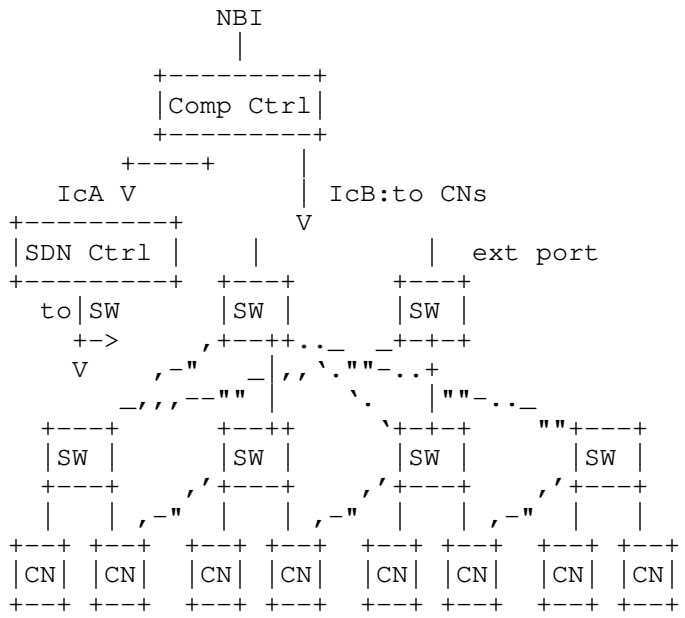


Figure 7: PoP DC Network with Compute Nodes (CN)

In turn, each single Compute Node (CN) may also have internal switching resources (see Figure 8). In a carrier environment, in order to meet data path requirements, allocation of compute node internal distributed resources (blades, CPU cores, etc.) may become equivalently important.

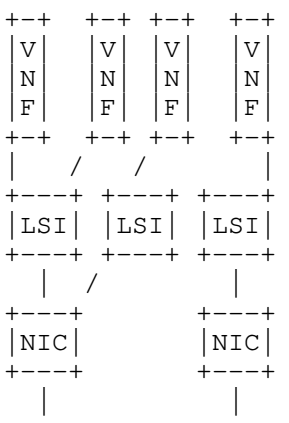


Figure 8: Compute Node with internal switching resource

Based on the recursion principles shown above and the complexity implied by separate interfaces for compute and network resources, one could imagine a recursive programmatic interface for joint compute, storage and network provisioning as depicted in Figure 9.

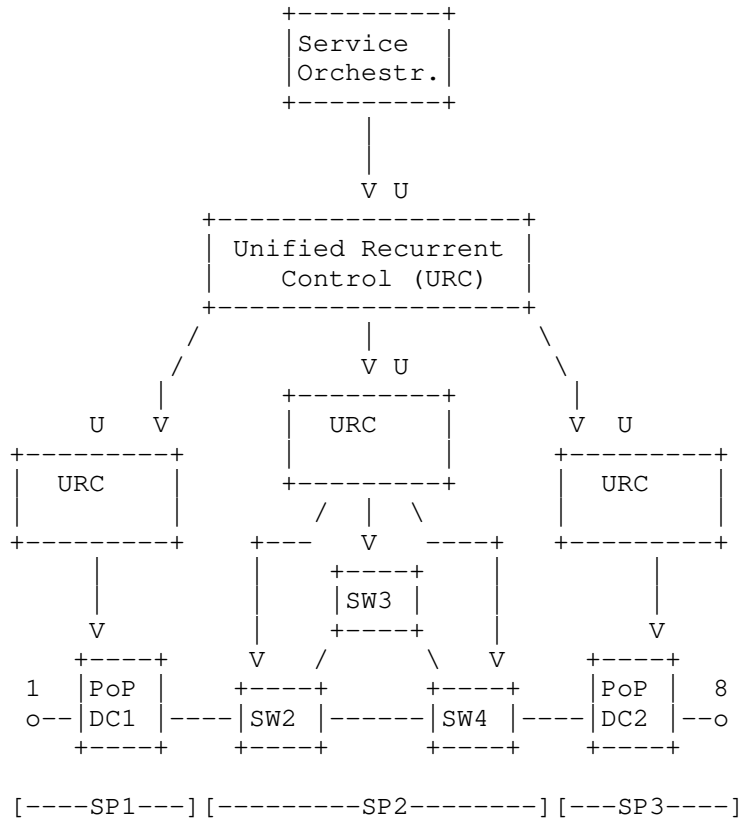


Figure 9: The RO Control Plane view considering a recursive programmatic interface for joint compute, storage and network provisioning

In Figure 9, Ic1, Ic2 and Ic3 of Figure 6 have been substituted by the recursive programmatic interface U to use for both compute and network resources and we find also the Unified Recurrent Control (URC), an element that performs both compute and network control and that can be used in a hierarchy structure.

Considering the use of the recursive programmatic interface U and the Unified Recurrent Control, the PoP DC Network structure with Compute Nodes view changes as reported in Figure 10.

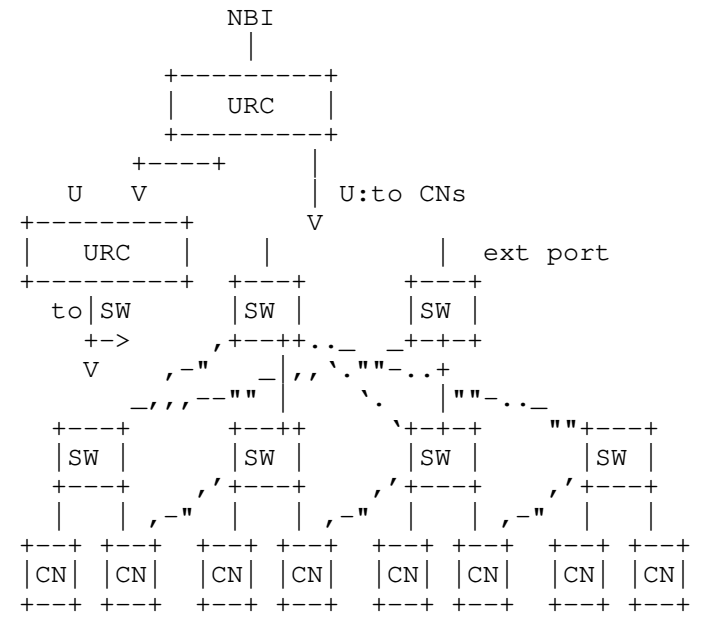


Figure 10: PoP DC Network with Compute Nodes (CN) considering the U interface and the URC element

4. Problem Statement

The motivational examples of Section 3 illustrate that almost always compute virtualization and network virtualization are tightly connected. In particular Figure, 3 shows that in a PoP DC there are not only compute resources (CNs) but also network resources (SWs), and so it illustrates that compute virtualization implicitly involves network virtualization unless we consider the unlikely scenario where dedicated network elements are used to interconnect the different virtual network functions implemented on the compute nodes (e.g.: to implement Flexible Service Chaining). On the other hand, considering a network scenario made not only of just pure SDN network elements (SWs) but also of compute resources (CNs) or SDN network nodes that are equipped also with compute resources (UNs), it is very likely that virtualized network resources, if offered to clients, imply virtualization of compute resources, unless we consider the unlikely scenario where dedicated compute resources are available for every virtualized network.

Furthermore, virtualization often leads to scenarios of recursions with clients redefining and reselling resources and services at different levels.

We argue that given the multi-level virtualization of compute, storage and network domains, automation of the corresponding resource provisioning could be more easily implemented by a recursive programmatic interface. Existing separated compute and network programming interfaces cannot easily provide such recursions and cannot always satisfy key requirement for multi-vendor, multi-technology and multi-provider interoperability environments. Therefore we foresee the necessity of a recursive programmatic interface for joint compute, storage and network provisioning.

5. Challenges

We summarize in this section the key questions and challenges, which we hope will initiate further discussions in the NFVRG community.

5.1. Orchestration

Firstly, as motivated in Section 3, 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?

Secondly, can such a joint compute, storage and network programmatic interface allow an automated resource orchestration similar to the recursive SDN architecture [ONF-SDN-ARCH]?

Thirdly, can such a joint compute, storage and network programmatic interface realize the functionally of an SFC Control plane [I-D.ietf-sfc-control-plane]? Our initial mapping and proof of concept experimentations are documented in [I-D.unify-sfc-control-plane-exp].

5.2. Resource description

Prerequisite for joint placement decisions of compute, storage and network is the adequate description of available resources. This means that the interfaces (IcA, IcB etc. in Figure 6 and Figure 7) are of bidirectional nature, exposing resources as well as reserving. 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., 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.

5.3. 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).

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)?

5.4. 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 crossing 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].

5.5. 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.ietf-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.

6. IANA Considerations

This memo includes no request to IANA.

7. Security Considerations

TBD

8. Acknowledgement

The authors would like to thank the UNIFY team for inspiring discussions and in particular Fritz-Joachim Westphal and Catalin Meirosu for their comments and suggestions on how to refine this draft.

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.

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>.

[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 (draft)", Jul. 2014, <http://docbox.etsi.org/ISG/NFV/Open/Latest_Drafts/NFV-MAN001v061-%20management%20and%20orchestration.pdf>.

[EU-5GPPP-Contract] 5G-PPP Association, "Contractual Arrangement: Setting up a Public- Private Partnership in the Area of Advance 5G Network Infrastructure for the Future Internet between the European Union and the 5G Infrastructure Association", Dec 2013, <http://5g-ppp.eu/contract/>.

[I-D.caszpe-nfvrg-orchestration-challenges] Carrozzo, G., Szabo, R., and K. Pentikousis, "Network Function Virtualization: Resource Orchestration Challenges", draft-caszpe-nfvrg-orchestration-challenges-00 (work in progress), November 2015.

[I-D.ietf-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-ietf-ippm-twamp-yang-00 (work in progress), March 2016.

[I-D.ietf-sfc-control-plane] Boucadair, M., "Service Function Chaining (SFC) Control Plane Components & Requirements", draft-ietf-sfc-control-plane-06 (work in progress), May 2016.

[I-D.ietf-sfc-problem-statement] Quinn, P. and T. Nadeau, "Service Function Chaining Problem Statement", draft-ietf-sfc-problem-statement-13 (work in progress), February 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-04 (work in progress), March 2016.

[I-D.unify-sfc-control-plane-exp]
Szabo, R. and B. Sonkoly, "A Multi-Domain Multi-Technology SFC Control Plane Experiment: A UNIFYed Approach", draft-unify-sfc-control-plane-exp-00 (work in progress), March 2016.

[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.

[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>.

[OpenStack]
The OpenStack project, "Openstack cloud software", 2014, <http://openstack.org>.

[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>.

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

[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

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

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

Andras Csaszar
Ericsson Research, Hungary
Irinyi Jozsef u. 4-20
Budapest 1117
Hungary

Email: andras.csaszar@ericsson.com
URI: http://www.ericsson.com/

Kostas Pentikousis
Travelping
Koernerstr. 7-10
Berlin 10785
Germany

Email: k.pentikousis@travelping.com

Mario Kind
Deutsche Telekom AG
Winterfeldtstr. 21
10781 Berlin
Germany

Email: mario.kind@telekom.de

Diego Daino
Telecom Italia
Via Guglielmo Reiss Romoli 274
10148 Turin
Italy

Email: diego.daino@telecomitalia.ite

Zu Qiang
Ericsson
8400, boul. Decarie
Ville Mont-Royal, QC 8400
Canada

Email: zu.qiang@ericsson.com
URI: http://www.ericsson.com/

Hagen Woesner
BISDN
Koernerstr. 7-10
Berlin 10785
Germany

Email: hagen.woesner@bisdn.de
URI: http://www.bisdn.de

NFVRG
Internet Draft
Intended status: Informational
Expires: January 2017

C. Meirosu
Ericsson
A. Manzalini
Telecom Italia
R. Steinert
SICS
G. Marchetto
Politecnico di Torino
K. Pentikousis
EICT
S. Wright
AT&T
P. Lynch
Ixia
W. John
Ericsson

July 8, 2016

DevOps for Software-Defined Telecom Infrastructures
draft-unify-nfvrg-devops-05.txt

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html

This Internet-Draft will expire on January 8, 2016.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Abstract

Carrier-grade network management was optimized for environments built with monolithic physical nodes and involves significant deployment, integration and maintenance efforts from network service providers. The introduction of virtualization technologies, from the physical layer all the way up to the application layer, however, invalidates several well-established assumptions in this domain. This draft opens the discussion in NFVRG about challenges related to transforming the telecom network infrastructure into an agile, model-driven environment for communication services. We take inspiration from data center DevOps on the simplification and automation of management processes for a telecom service provider software-defined infrastructure (SDI). A number of challenges associated with operationalizing DevOps principles at scale in software-defined telecom networks are identified in relation to three areas related to key programmable management processes.

Table of Contents

1. Introduction

Carrier-grade network management was developed as an incremental solution once a particular network technology matured and came to be deployed in parallel with legacy technologies. This approach requires significant integration efforts when new network services are launched. Both centralized and distributed algorithms have been developed in order to solve very specific problems related to configuration, performance and fault management. However, such algorithms consider a network that is by and large functionally static. Thus, management processes related to introducing new or maintaining functionality are complex and costly due to significant efforts required for verification and integration.

Network virtualization, by means of Software-Defined Networking (SDN) and Network Function Virtualization (NFV), creates an environment where network functions are no longer static or strictly embedded in physical boxes deployed at fixed points. The virtualized network is dynamic and open to fast-paced innovation enabling efficient network management and reduction of operating cost for network operators. A significant part of network capabilities are expected to become available through interfaces that resemble the APIs widespread within datacenters instead of the traditional telecom means of management such as the Simple Network Management Protocol, Command Line Interfaces or CORBA. Such an API-based approach, combined with the programmability offered by SDN interfaces [RFC7426], open opportunities for handling infrastructure, resources, and Virtual Network Functions (VNFs) as code, employing techniques from software engineering.

The efficiency and integration of existing management techniques in virtualized and dynamic network environments are limited, however. Monitoring tools, e.g. based on simple counters, physical network

taps and active probing, do not scale well and provide only a small part of the observability features required in such a dynamic environment. Although huge amounts of monitoring data can be collected from the nodes, the typical granularity is rather static and coarse and management bandwidths may be limited. Debugging and troubleshooting techniques developed for software-defined environments are a research topic that has gathered interest in the research community in the last years. Still, it is yet to be explored how to integrate them into an operational network management system. Moreover, research tools developed in academia (such as NetSight [H2014], OFRewind [W2011], FlowChecker [S2010], etc.) were limited to solving very particular, well-defined problems, and oftentimes are not built for automation and integration into carrier-grade network operations workflows. As the virtualized network functions, infrastructure software and infrastructure hardware become more dynamic [NFVSWA], the monitoring, management and testing approaches also need to change.

The topics at hand have already attracted several standardization organizations to look into the issues arising in this new environment. For example, IETF working groups have activities in the area of OAM and Verification for Service Function Chaining [I-D.aldrin-sfc-oam-framework] [I-D.lee-sfc-verification] for Service Function Chaining. At IRTF, [RFC7149] asks a set of relevant questions regarding operations of SDNs. The ETSI NFV ISG defines the MANO interfaces [NFVMANO], and TMForum investigates gaps between these interfaces and existing specifications in [TR228]. The need for programmatic APIs in the orchestration of compute, network and storage resources is discussed in [I-D.unify-nfvrg-challenges].

From a research perspective, problems related to operations of software-defined networks are in part outlined in [SDNsurvey] and research referring to both cloud and software-defined networks are discussed in [D4.1].

The purpose of this first version of this document is to act as a discussion opener in NFVRG by describing a set of principles that are relevant for applying DevOps ideas to managing software-defined telecom network infrastructures. We identify a set of challenges related to developing tools, interfaces and protocols that would support these principles and how can we leverage standard APIs for simplifying management tasks.

2. Software-Defined Telecom Infrastructure: Roles and DevOps principles

There is no single list of core principles of DevOps, but it is generally recognized as encompassing:

- Iterative development / Incremental feature content
- Continuous deployment
- Automated processes
- Holistic/Systemic views of development and deployment/ operation.

With Deployment/ Operations becoming increasingly linked with software development, and business needs driving more rapid deployments, agile methodologies are assumed as a basis for DevOps. Agile methods used in many software focused companies are focused on releasing small interactions of code to implement VNFs with high velocity and high quality into a production environment. Similarly, Service providers are interested to release incremental improvements in the network services that they create from virtualized network functions. The cycle time for DevOps as applied in many open source projects is on the order of one quarter year or 13 weeks.

The code needs to undergo a significant amount of automated testing and verification with pre-defined templates in a realistic setting. From the point of view of software defined telecom infrastructure management, the of the network and service configuration is expected to continuously evolve as result of network policy decomposition and refinement, service evolution, the updates, failovers or re-configuration of virtual functions, additions/upgrades of new infrastructure resources (e.g. whiteboxes, fibers). When troubleshooting the cause of unexpected behavior, fine-grained visibility onto all resources supporting the virtual functions (either compute, or network-related) is paramount to facilitating fast resolution times. While compute resources are typically very well covered by debugging and profiling toolsets based on many years of advances in software engineering, programmable network resources are a still a novelty and tools exploiting their potential are scarce.

2.1. Service Developer Role

We identify two dimensions of the "developer" role in software-defined infrastructure (SDI). The network service to be developed is captured in a network service descriptor (e.g. [IFA014]). One dimension relates to determining which high-level functions should be part of a particular service, deciding what logical interconnections are needed between these blocks and defining a set of high-level constraints or goals related to parameters that define, for instance, a Service Function Chain. This could be determined by the product owner for a particular family of services offered by a telecom provider. Or, it might be a key account representative that adapts an existing service template to the requirements of a particular customer by adding or removing a small number of functional entities. We refer to this person as the Service Developer and for simplicity (access control, training on technical background, etc.) we consider the role to be internal to the telecom provider.

2.2. VNF Developer role

Another dimension of the "developer" role is a person that writes the software code for a new virtual network function (VNF). The VNF then needs to be delivered as a package (e.g.[IFA011]) that includes various metadata for ingestion/integration into some service. Note that a VNF may span multiple virtual machines to support design objectives (e.g. for reliability or scalability). Depending on the actual VNF being developed, this person might be internal or external (e.g. a traditional equipment vendor) to the telecom provider. We refer to them as VNF Developers.

2.3. System Integrator role

The System Integrator role is to some extent similar to the Service Developer: people in this role need to identify the components of the system to be delivered. However, for the Service Developer, the service components are pre-integrated meaning that they have the right interfaces to interact with each other. In contrast, the Systems Integrator needs to develop the software that makes the system components interact with each other. As such, the Systems Integrator role combines aspects of the Developer roles and adds yet another dimension to it. Compared to the other Developer roles, the System Integrator might face additional challenges due to the fact that they might not have access to the source code of some of the components. This limits for example how fast they could address issues with components to be integrated, as well as uneven workload depending on the release granularity of the different components that need to be integrated. Some system integration activities may take

place on an industry basis in collaborative communities (e.g. OPNFV.org).

2.4. Network service Operator role

The role of a Network Service Operator is to ensure that the deployment processes were successful and a set of performance indicators associated to a particular network service are met. The network service is supported on infrastructure specific set of infrastructure resources that may be owned and operated by that Network Service Operator, or provided under contract from some other infrastructure service provider. .

2.5. Customer role

A Customer contracts a telecom operator to provide one or more services. In SDI, the Customer may communicate with the provider in real time through an online portal. From the customer perspective, such portal interfaces become part of the service definition just like the data transfer aspects of the service. Compared to the Service Developer, the Customer is external to the operator and may define changes to their own service instance only in accordance to policies defined by the Service Developer. In addition to the usual per-service utilization statistics, in SDI the portal may enable the customer to trigger certain performance management or troubleshooting tools for the service. This, for example, enables the Customer to determine whether the root cause of certain error or degradation condition that they observe is located in the telecom operator domain or not and may facilitate the interaction with the customer support teams.

2.6. DevOps Principles

In line with the generic DevOps concept outlined in [DevOpsP], we consider that these four principles as important for adapting DevOps ideas to SDI:

* Automated processes: Deploy with repeatable, reliable processes: Service and VNF Developers should be supported by automated build, orchestrate and deploy processes that are identical in the development, test and production environments. Such processes need to be made reliable and trusted in the sense that they should reduce the chance of human error and provide visibility at each stage of the process, as well as have the possibility to enable manual interactions in certain key stages.

* Holistis/systemic view: Develop and test against production-like systems: both Service Developers and VNF Developers need to have the opportunity to verify and debug their respective SDI code in systems that have characteristics which are very close to the production environment where the code is expected to be ultimately deployed. Customizations of Service Function Chains or VNFs could thus be released frequently to a production environment in compliance with policies set by the Operators. Adequate isolation and protection of the services active in the infrastructure from services being tested or debugged should be provided by the production environment.

* Continuous: Monitor and validate operational quality: Service Developers, VNF Developers and Operators must be equipped with tools, automated as much as possible, that enable to continuously monitor the operational quality of the services deployed on SDI. Monitoring tools should be complemented by tools that allow verifying and validating the operational quality of the service in line with established procedures which might be standardized (for example, Y.1564 Ethernet Activation [Y1564]) or defined through best practices specific to a particular telecom operator.

* Iterative/Incremental: Amplify development cycle feedback loops: An integral part of the DevOps ethos is building a cross-cultural environment that bridges the cultural gap between the desire for continuous change by the Developers and the demand by the Operators for stability and reliability of the infrastructure. Feedback from customers is collected and transmitted throughout the organization. From a technical perspective, such cultural aspects could be addressed through common sets of tools and APIs that are aimed at providing a shared vocabulary for both Developers and Operators, as well as simplifying the reproduction of problematic situations in the development, test and operations environments.

Network operators that would like to move to agile methods to deploy and manage their networks and services face a different environment compared to typical software companies where simplified trust relationships between personnel are the norm. In software companies, it is not uncommon that the same person may be rotating between different roles. In contrast, in a telecom service provider, there are strong organizational boundaries between suppliers (whether in Developer roles for network functions, or in Operator roles for outsourced services) and the carrier’s own personnel that might also take both Developer and Operator roles. Extending DevOps principles across strong organizational boundaries e.g. through co-creation or collaborative development in open source communities) may be a commercial challenge rather than a technical issue.

3. Continuous Integration

Software integration is the process of bringing together the software component subsystems into one software system, and ensuring that the subsystems function together as a system. Software integration can apply regardless of the size of the software components. The objective of Continuous Integration is to prevent integration problems close to the expected release of a software development project into a production (operations) environment. Continuous Integration is therefore closely coupled with the notion of DevOps as a mechanism to ease the transition from development to operations.

Continuous integration may result in multiple builds per day. It is also typically used in conjunction with test driven development approaches that integrate unit testing into the build process. The unit testing is typically automated through build servers. Such servers may implement a variety of additional static and dynamic tests as well as other quality control and documentation extraction functions. The reduced cycle times of continuous enable improved software quality by applying small efforts frequently.

Continuous Integration applies to developers of VNF as they integrate the components that they need to deliver their VNF. The VNFs may contain components developed by different teams within the VNF Provider, or may integrate code developed externally - e.g. in commercial code libraries or in open source communities.

Service developers also apply continuous integration in the development of network services. Network services are comprised of various aspects including VNFs and connectivity within and between them as well as with various associated resource authorizations. The components of the networks service are all dynamic, and largely represented by software that must be integrated regularly to maintain consistency.

Some of the software components that Service Developers integrate may be sourced from VNF Providers or from open source communities. Service Developers and Network Service Operators are increasingly motivated to engage with open Source communities [OSandS]. Open source interfaces supported by open source communities may be more useful than traditional paper interface specifications. Even where Service Providers are deeply engaged in the open source community (e.g. OPNFV) many service providers may prefer to obtain the code through some software provider as a business practice. Such software providers have the same interests in software integration as other

VNF providers. An open source integration community (e.g. OPNFV) may resolve common integration issues across the industry reducing the need for integration issue resolution specific to particular integrators.

4. Continuous Delivery

The practice of Continuous Delivery extends Continuous Integration by ensuring that the software (either a VNF code or code for SDI) checked in on the mainline is always in a user deployable state and enables rapid deployment by those users. For critical systems such as telecommunications networks, Continuous Delivery may require the advantage of including a manual trigger before the actual deployment in the live system, compared to the Continuous Deployment methodology which is also part of DevOps processes in software companies.

Automated Continuous deployment systems in may exceed 10 updates per day. Assuming an integration of 100 components, each with an average time to upgrade of 180 days then deployments on the order of every 1.8 days might be expected. The telecom infrastructure is also very distributed - consider the case of cloud RAN use cases where the number of locations for deployment is of the order of the number of cell tower locations (~10^4..10^6). Deployments may need to be incremental across the infrastructure to reduce the risk of large-scale failures. Conversely, there may need to be rapid rollbacks to prior stable deployment configurations in the event of significant failures.

5. Consistency, Availability and Partitioning Challenges

The CAP theorem [CAP] states that any networked shared-data system can have at most two of following three properties: 1) Consistency (C) equivalent to having a single up-to-date copy of the data; 2) high Availability (A) of that data (for updates); and 3) tolerance to network Partitions (P).

Looking at a telecom SDI as a distributed computational system (routing/forwarding packets can be seen as a computational problem), just two of the three CAP properties will be possible at the same time. The general idea is that 2 of the 3 have to be chosen. CP favor consistency, AP favor availability, CA there are no partition. This has profound implications for technologies that need to be developed in line with the "deploy with repeatable, reliable processes"

principle for configuring SDI states. Latency or delay and partitioning properties are closely related, and such relation becomes more important in the case of telecom service providers where Devs and Ops interact with widely distributed infrastructure. Limitations of interactions between centralized management and distributed control need to be carefully examined in such environments. Traditionally connectivity was the main concern: C and A was about delivering packets to destination. The features and capabilities of SDN and NFV are changing the concerns: for example in SDN, control plane Partitions no longer imply data plane Partitions, so A does not imply C. In practice, CAP reflects the need for a balance between local/distributed operations and remote/centralized operations.

Furthermore to CAP aspects related to individual protocols, interdependencies between CAP choices for both resources and VNFs that are interconnected in a forwarding graph need to be considered. This is particularly relevant for the "Monitor and Validate Operational Quality" principle, as apart from transport protocols, most OAM functionality is generally configured in processes that are separated from the configuration of the monitored entities. Also, partitioning in a monitoring plane implemented through VNFs executed on compute resources does not necessarily mean that the dataplane of the monitored VNF was partitioned as well.

6. Stability and Real-Time Change Challenges

The dimensions, dynamicity and heterogeneity of networks are growing continuously. Monitoring and managing the network behavior in order to meet technical and business objectives is becoming increasingly complicated and challenging, especially when considering the need of predicting and taming potential instabilities.

In general, instability in networks may have primary effects both jeopardizing the performance and compromising an optimized use of resources, even across multiple layers: in fact, instability of end-to-end communication paths may depend both on the underlying transport network, as well as the higher level components specific to flow control and dynamic routing. For example, arguments for introducing advanced flow admission control are essentially derived from the observation that the network otherwise behaves in an inefficient and potentially unstable manner. Even with resources over provisioning, a network without an efficient flow admission control has instability regions that can even lead to congestion collapse in certain configurations. Another example is the instability which is

characteristic of any dynamically adaptive routing system. Routing instability, which can be (informally) defined as the quick change of network reachability and topology information, has a number of possible origins, including problems with connections, router failures, high levels of congestion, software configuration errors, transient physical and data link problems, and software bugs.

As a matter of fact, the states monitored and used to implement the different control and management functions in network nodes are governed by several low-level configuration commands. There are several dependencies among these states and the logic updating the states in real time (most of which are not synchronized automatically). Normally, high-level network goals (such as the connectivity matrix, load-balancing, traffic engineering goals, survivability requirements, etc) are translated into low-level configuration commands (mostly manually) individually executed on the network elements (e.g., forwarding table, packet filters, link-scheduling weights, and queue-management parameters, as well as tunnels and NAT mappings). Network instabilities due to configuration errors can spread from node to node and propagate throughout the network.

DevOps in the data center is a source of inspiration regarding how to simplify and automate management processes for software-defined infrastructure. Although the low-level configuration could be automated by DevOps tools such as CFEngine [C2015], Puppet [P2015] and Ansible [A2015], the high-level goal translation towards tool-specific syntax is still a manual process. In addition, while carrier-grade configuration tools using the NETCONF protocol support complex atomic transaction management (which reduces the potential for instability), Ansible requires third-party components to support rollbacks and the Puppet transactions are not atomic.

As a specific example, automated configuration functions are expected to take the form of a "control loop" that monitors (i.e., measures) current states of the network, performs a computation, and then reconfigures the network. These types of functions must work correctly even in the presence of failures, variable delays in communicating with a distributed set of devices, and frequent changes in network conditions. Nevertheless cascading and nesting of automated configuration processes can lead to the emergence of non-linear network behaviors, and as such sudden instabilities (i.e. identical local dynamic can give rise to widely different global dynamics).

7. Observability Challenges

Monitoring algorithms need to operate in a scalable manner while providing the specified level of observability in the network, either for operation purposes (Ops part) or for debugging in a development phase (Dev part). We consider the following challenges:

* Scalability – relates to the granularity of network observability, computational efficiency, communication overhead, and strategic placement of monitoring functions.

* Distributed operation and information exchange between monitoring functions – monitoring functions supported by the nodes may perform specific operations (such as aggregation or filtering) locally on the collected data or within a defined data neighborhood and forward only the result to a management system. Such operation may require modifications of existing standards and development of protocols for efficient information exchange and messaging between monitoring functions. Different levels of granularity may need to be offered for the data exchanged through the interfaces, depending on the Dev or Ops role. Modern messaging systems, such as Apache Kafka [AK2015], widely employed in datacenter environments, were optimized for messages that are considerably larger than reading a single counter value (typical SNMP GET call usage) – note the throughput vs record size from [K2014]. It is also debatable to what extent properties such as message persistence within the bus are needed in a carrier environment, where MIBs practically offer already a certain level of persistence of management data at the node level. Also, they require the use of IP addressing which might not be needed when the monitored data is consumed by a function within the same node.

* Common communication channel between monitoring functions and higher layer entities (orchestration, control or management systems) – a single communication channel for configuration and measurement data of diverse monitoring functions running on heterogeneous hard- and software environments. In telecommunication environments, infrastructure assets span not only large geographical areas, but also a wide range of technology domains, ranging from CPEs, access-, aggregation-, and transport networks, to datacenters. This heterogeneity of hard- and software platforms requires higher layer entities to utilize various parallel communication channels for either configuration or data retrieval of monitoring functions within these technology domains. To address automation and advances in monitoring programmability, software defined telecommunication infrastructures would benefit from a single flexible communication channel, thereby supporting the dynamicity of virtualized environments. Such a channel should ideally support propagation of

configuration, signalling, and results from monitoring functions; carrier-grade operations in terms of availability and multi-tenant features; support highly distributed and hierarchical architectures, keeping messages as local as possible; be lightweight, topology independent, network address agnostic; support flexibility in terms of transport mechanisms and programming language support. Existing popular state-of-the-art message queuing systems such as RabbitMQ [R2015] fulfill many of these requirements. However, they utilize centralized brokers, posing a single point-of-failure and scalability concerns within vastly distributed NFV environment. Furthermore, transport support is limited to TCP/IP. ZeroMQ [Z2015] on the other hard lacks any advanced features for carrier-grade operations, including high-availability, authentication, and tenant isolation.

* Configurability and conditional observability - monitoring functions that go beyond measuring simple metrics (such as delay, or packet loss) require expressive monitoring annotation languages for describing the functionality such that it can be programmed by a controller. Monitoring algorithms implementing self-adaptive monitoring behavior relative to local network situations may employ such annotation languages to receive high-level objectives (KPIs controlling tradeoffs between accuracy and measurement frequency, for example) and conditions for varying the measurement intensity. Steps in this direction were taken by the DevOps tools such as Splunk [S2015], whose collecting agent has the ability to load particular apps that in turn access specific counters or log files. However, such apps are tool specific and may also require deploying additional agents that are specific to the application, library or infrastructure node being monitored. Choosing which objects to monitor in such environment means deploying a tool-specific script that configures the monitoring app.

* Automation - includes mapping of monitoring functionality from a logical forwarding graph to virtual or physical instances executing in the infrastructure, as well as placement and re-placement of monitoring functionality for required observability coverage and configuration consistency upon updates in a dynamic network environment. Puppet [P2015] manifests or Ansible [A2015] playbooks could be used for automating the deployment of monitoring agents, for example those used by Splunk [S2015]. However, both manifests and playbooks were designed to represent the desired system configuration snapshot at a particular moment in time - they would now need to be generated automatically by the orchestration tools instead of a DevOps person.

* Actionable data

Data produced by observability tools could be utilized in a wide category of processes, ranging from billing and dimensioning to real-time troubleshooting and optimization. In order to allow for data-driven automated decisions and actuations based on these decisions, the data needs to be actionable. We define actionable data as being representative for a particular context or situation and an adequate input towards a decision. Ensuring actionable data is challenging in a number of ways, including: defining adaptive correlation and sampling windows, filtering and aggregation methods that are adapted or coordinated with the actual consumer of the data, and developing analytical and predictive methods that account for the uncertainty or incompleteness of the data.

* Data Virtualization

Data is key in helping both Developers and Operators perform their tasks. Traditional Network Management Systems were optimized for using one database that contains the master copy of the operational statistics and logs of network nodes. Ensuring access to this data from across the organization is challenging because strict privacy and business secrets need to be protected. In DevOps-driven environments, data needs to be made available to Developers and their test environments. Data virtualization collectively defines a set of technologies that ensure that restricted copies of the partial data needed for a particular task may be made available while enforcing strict access control. Further than simple access control, data virtualization needs to address scalability challenges involved in copying large amounts of operational data as well as automatically disposing of it when the task authorized for using it has finished.

8. Verification Challenges

Enabling ongoing verification of code is an important goal of continuous integration as part of the data center DevOps concept. In a telecom SDI, service definitions, decompositions and configurations need to be expressed in machine-readable encodings. For example, configuration parameters could be expressed in terms of YANG data models. However, the infrastructure management layers (such as Software-Defined Network Controllers and Orchestration functions) might not always export such machine-readable descriptions of the runtime configuration state. In this case, the management layer itself could be expected to include a verification process that has the same challenges as the stand-alone verification processes we outline later in this section. In that sense, verification can be considered as a set of features providing gatekeeper functions to

verify both the abstract service models and the proposed resource configuration before or right after the actual instantiation on the infrastructure layer takes place.

A verification process can involve different layers of the network and service architecture. Starting from a high-level verification of the customer input (for example, a Service Graph as defined in [I-D.unify-nfvrg-challenges]), the verification process could go more in depth to reflect on the Service Function Chain configuration. At the lowest layer, the verification would handle the actual set of forwarding rules and other configuration parameters associated to a Service Function Chain instance. This enables the verification of more quantitative properties (e.g. compliance with resource availability), as well as a more detailed and precise verification of the abovementioned topological ones. Existing SDN verification tools could be deployed in this context, but the majority of them only operate on flow space rules commonly expressed using OpenFlow syntax.

Moreover, such verification tools were designed for networks where the flow rules are necessary and sufficient to determine the forwarding state. This assumption is valid in networks composed only by network functions that forward traffic by analyzing only the packet headers (e.g. simple routers, stateless firewalls, etc.). Unfortunately, most of the real networks contain active network functions, represented by middle-boxes that dynamically change the forwarding path of a flow according to function-local algorithms and an internal state (that is based on the received packets), e.g. load balancers, packet marking modules and intrusion detection systems. The existing verification tools do not consider active network functions because they do not account for the dynamic transformation of an internal state into the verification process.

Defining a set of verification tools that can account for active network functions is a significant challenge. In order to perform verification based on formal properties of the system, the internal states of an active (virtual or not) network function would need to be represented. Although these states would increase the verification process complexity (e.g., using simple model checking would not be feasible due to state explosion), they help to better represent the forwarding behavior in real networks. A way to address this challenge is by attempting to summarize the internal state of an active network function in a way that allows for the verification process to finish within a reasonable time interval.

9. Testing Challenges

Testing in an NFV environment does impact the methodology used. The main challenge is the ability to isolate the Device Under Test (DUT). When testing physical devices, which are dedicated to a specific function, isolation of this function is relatively simple: isolate the DUT by surrounding it with emulations from test devices. This achieves isolation of the DUT, in a black box fashion, for any type of testing. In an NFV environment, the DUT become a component of a software infrastructure which can't be isolated. For example, testing a VNF can't be achieved without the presence if the NFVI and MANO components. In addition, the NFVI and MANO components can greatly influence the behavior and the performance of the VNF under test.

With this in mind, in NFV, the isolation of the DUT becomes a new concept: the VNF Under Test (VUT) becomes part of an environment that consists of the rest of the necessary architecture components (the test environment). In the previous example, the VNF becomes the VUT, while the MANO and NFVI become the test environment. Then, isolation of the VUT becomes a matter of configuration management, where the configuration of the test environment is kept fixed for each test of the VUT. So the MANO policies for instantiation, scaling, and placement, as well as the NFVI parameters such as HW used, CPU pinning, etc must remained fixed for each iterative test of the VNF. Only by keeping the configurations constant can the VNF tests can be compared to each other. If any test environment configurations are changed between tests, the behavior of the VNF can be impacted, thus negating any comparison of the results.

Of course, there are instances of testing where the inverse is desired: the configuration of the test environment is changed between each test, while the VNF configuration is kept constant. As an example, this type of methodology would be used in order to discover the optimum configuration of the NFVI for a particular VNF workload. Another similar but daunting challenge is the introduction of co-located tenants in the same environment as the VNF under test. The workload on these "neighbors" can greatly influence the behavior and performance of the VNF under test, but the test itself is invaluable to understand the impact of such a configuration.

Another challenge is the usage of test devices (traffic generator, emulator) that share the same infrastructure as the VNF under test. This can create a situation as above, where the neighbor competes for resources with the VUT itself, which can really negate test results. If a test architecture such as this is necessary (testing east-west traffic, for example), then care must be taken to configure the test devices such as they are isolated from the SUT in terms of allowed

resources, and that they don't impact the SUT's ability to acquire resources to operate in all conditions.

NFV offers new features that didn't exist as such previously, or modifies existing mechanisms. Examples of new features are dynamic scaling of VNFs and network services (NS), standardized acceleration mechanisms and the presence of the virtualization layer, which includes the vSwitch. An example mechanism which changes with NFV how fault detection and fault recovery are handled. Fault recovery could now be handled by MANO in such a way to invoke mechanisms such as live migration or snapshots in order to recover the state of a VNF and restore operation quickly. While the end results are expected to be the same as before, since the mechanism is very different, rigorous testing is highly recommended to validate those results.

Dynamic scaling of VNFs is a new concept in NFV. VNFs that require more resources will have them dynamically allocated on demand, and then subsequently released when not needed anymore. This is clearly a benefit arising from SDI. For each type of VNF, specific metrics will be used as input to conditions that will trigger a scaling operation, orchestrated by MANO. Testing this mechanism requires a methodology tailored to the specific operation of the VNF, in order to properly reach the monitored metrics and exercise the conditions leading to a scaling trigger. For example, a firewall VNF will be triggered for scaling on very different metrics than a 3GPP MME. Both VNFs accomplish different functions. Since there will normally be a collection of metrics that are monitored in order to trigger a scaling operation, the testing methodology must be constructed in such a way as to address all combinations of those metrics. Metrics for a particular VNF may include sessions, session instantiations/second, throughput, etc. These metrics will be observed in relation to the given resources for the VNF.

10. Programmable management

The ability to automate a set of actions to be performed on the infrastructure, be it virtual or physical, is key to productivity increases following the application of DevOps principles. Previous sections in this document touched on different dimensions of programmability:

- Section 5 approached programmability in the context of developing new capabilities for monitoring and for dynamically setting configuration parameters of deployed monitoring functions

- Section 7 reflected on the need to determine the correctness of actions that are to be inflicted on the infrastructure as result of executing a set of high-level instructions

- Section 8 considered programmability in the perspective of an interface to facilitate dynamic orchestration of troubleshooting steps towards building workflows and for reducing the manual steps required in troubleshooting processes

We expect that programmable network management - along the lines of [RFC7426] - will draw more interest as we move forward. For example, in [I-D.unify-nfvrg-challenges], the authors identify the need for presenting programmable interfaces that accept instructions in a standards-supported manner for the Two-way Active Measurement Protocol (TWAMP)TWAMP protocol. More specifically, an excellent example in this case is traffic measurements, which are extensively used today to determine SLA adherence as well as debug and troubleshoot pain points in service delivery. TWAMP is both widely implemented by all established vendors and deployed by most global operators. However, TWAMP management and control today relies solely on diverse and proprietary tools provided by the respective vendors of the equipment. For large, virtualized, and dynamically instantiated infrastructures where network functions are placed according to orchestration algorithms proprietary mechanisms for managing TWAMP measurements have severe limitations. For example, today's TWAMP implementations are managed by vendor-specific, typically command-line interfaces (CLI), which can be scripted on a platform-by-platform basis. As a result, although the control and test measurement protocols are standardized, their respective management is not. This hinders dramatically the possibility to integrate such deployed functionality in the SP-DevOps concept. In this particular case, recent efforts in the IPPM WG [I-D.cmzrjp-ippm-twamp-yang] aim to define a standard TWAMP data model and effectively increase the programmability of TWAMP deployments in the future.

Data center DevOps tools, such as those surveyed in [D4.1], developed proprietary methods for describing and interacting through interfaces with the managed infrastructure. Within certain communities, they became de-facto standards in the same way particular CLIs became de-facto standards for Internet professionals. Although open-source components and a strong community involvement exists, the diversity of the new languages and interfaces creates a burden for both vendors in terms of choosing which ones to prioritize for support, and then developing the functionality and operators that determine what fits best for the requirements of their systems.

11. Security Considerations

DevOps principles are typically practiced within the context of a single organization ie a single trust domain. Extending DevOps practices across strong organizational boundaries (e.g. between commercial organizations) requires consideration of additional threat models. Additional validation procedures may be required to ingest and accept code changes arising from outside an organization.

12. IANA Considerations

This memo includes no request to IANA.

13. References

13.1. Informative References

[NFVMANO] ETSI, "Network Function Virtualization (NFV) Management and Orchestration V0.6.1 (draft)", Jul. 2014

[I-D.aldrin-sfc-oam-framework] S. Aldrin, R. Pignataro, N. Akiya. "Service Function Chaining Operations, Administration and Maintenance Framework", draft-aldrin-sfc-oam-framework-02, (work in progress), July 2015.

[I-D.lee-sfc-verification] S. Lee and M. Shin. "Service Function Chaining Verification", draft-lee-sfc-verification-00, (work in progress), February 2014.

[RFC7426] E. Haleplidis (Ed.), K. Pentikousis (Ed.), S. Denazis, J. Hadi Salim, D. Meyer, and O. Koufopavlou, "Software Defined Networking (SDN): Layers and Architecture Terminology", RFC 7426, January 2015

[RFC7149] M. Boucadair and C Jaquenet. "Software-Defined Networking: A Perspective from within a Service Provider Environment", RFC 7149, March 2014.

[TR228] TMForum Gap Analysis Related to MANO Work. TR228, May 2014

[I-D.unify-nfvrg-challenges] R. Szabo et al. "Unifying Carrier and Cloud Networks: Problem Statement and Challenges", draft-unify-nfvrg-challenges-03 (work in progress), October 2016

[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.

[D4.1] W. John et al. D4.1 Initial requirements for the SP-DevOps concept, universal node capabilities and proposed tools, August 2014.

[SDNsurvey] D. Kreutz, F. M. V. Ramos, P. Verissimo, C. Esteve Rothenberg, S. Azodolmolky, S. Uhlig. "Software-Defined Networking: A Comprehensive Survey." To appear in proceedings of the IEEE, 2015.

[DevOpsP] "DevOps, the IBM Approach" 2013. [Online].

[Y1564] ITU-R Recommendation Y.1564: Ethernet service activation test methodology, March 2011

[CAP] E. Brewer, "CAP twelve years later: How the "rules" have changed", IEEE Computer, vol.45, no.2, pp.23,29, Feb. 2012.

[H2014] N. Handigol, B. Heller, V. Jeyakumar, D. Mazieres, N. McKeown; "I Know What Your Packet Did Last Hop: Using Packet Histories to Troubleshoot Networks", In Proceedings of the 11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14), pp.71-95

[W2011] A. Wundsam, D. Levin, S. Seetharaman, A. Feldmann; "OFRewind: Enabling Record and Replay Troubleshooting for Networks". In Proceedings of the Usenix Anual Technical Conference (Usenix ATC '11), pp 327-340

[S2010] E. Al-Shaer and S. Al-Haj. "FlowChecker: configuration analysis and verification of federated Openflow infrastructures" In Proceedings of the 3rd ACM workshop on Assurable and usable security configuration (SafeConfig '10). Pp. 37-44

[OSandS] S. Wright, D. Druta, "Open Source and Standards: The Role of Open Source in the Dialogue between Research and Standardization" Globecom Workshops (GC Wkshps), 2014 , pp.650,655, 8-12 Dec. 2014

[C2015] CFEngine. Online: http://cfengine.com/product/what-is-cfengine/, retrieved Sep 23, 2015.

[P2015] Puppet. Online: http://puppetlabs.com/puppet/what-is-puppet, retrieved Sep 23, 2015.

[A2015] Ansible. Online: http://docs.ansible.com/ , retrieved Sep 23, 2015.

[AK2015] Apache Kafka. Online: http://kafka.apache.org/documentation.html, retrieved Sep 23, 2015.

[S2015] Splunk. Online: http://www.splunk.com/en_us/products/splunk-light.html , retrieved Sep 23, 2015.

[K2014] J. Kreps. Benchmarking Apache Kafka: 2 Million Writes Per Second (On Three Cheap Machines). Online: https://engineering.linkedin.com/kafka/benchmarking-apache-kafka-2-million-writes-second-three-cheap-machines, retrieved Sep 23, 2015.

[R2015] RabbitMQ. Online: https://www.rabbitmq.com/ , retrieved Oct 13, 2015

[IFA014] ETSI, Network Functions Virtualisation (NFV); Management and Orchestration Network Service Templates Specification , DGS/NFV-IFA014, Work In Progress

[IFA011] ETSI, Network Functions Virtualisation (NFV); Management and Orchestration; VNF Packaging Specification, DGS/NFV-IFA011, Work in Progress

[NFVSWA] ETSI, Network functions Virtualisation; Virtual Network Functions Architecture, GS NFV-SWA 001 v1.1.1 (2014)

[Z2015] ZeroMQ. Online: http://zeromq.org/ , retrieved Oct 13, 2015

14. Contributors to earlier versions

J. Kim (Deutsche Telekom), S. Sharma (iMinds), I. Papafili (OTE)

15. Acknowledgments

The research leading to these results has received funding from the European Union Seventh Framework Programme FP7/2007-2013 under grant agreement no. 619609 - the UNIFY project. 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.

We would like to thank in particular the UNIFY WP4 contributors, the internal reviewers of the UNIFY WP4 deliverables and Russ White and Ramki Krishnan for their suggestions.

This document was prepared using 2-Word-v2.0.template.dot.

16. Authors' Addresses

Catalin Meirosu
Ericsson Research
S-16480 Stockholm, Sweden
Email: catalin.meirosu@ericsson.com

Antonio Manzalini
Telecom Italia
Via Reiss Romoli, 274
10148 - Torino, Italy
Email: antonio.manzalini@telecomitalia.it

Rebecca Steinert
SICS Swedish ICT AB
Box 1263, SE-16429 Kista, Sweden
Email: rebste@sics.se

Guido Marchetto
Politecnico di Torino
Corso Duca degli Abruzzi 24
10129 - Torino, Italy
Email: guido.marchetto@polito.it

Kostas Pentikousis
Travelping GmbH
Koernerstrasse 7-10
Berlin 10785
Germany
Email: k.pentikousis@travelping.com

Steven Wright
AT&T Services Inc.
1057 Lenox Park Blvd NE, STE 4D28
Atlanta, GA 30319
USA
Email: sw3588@att.com

Pierre Lynch
Ixia
800 Perimeter Park Drive, Suite A
Morrisville, NC 27560

USA
Email: plynch@ixiacom.com

Wolfgang John
Ericsson Research
S-16480 Stockholm, Sweden
Email: wolfgang.john@ericsson.com

Internet Research Task Force (IRTF)
Internet Draft
Intended status: Informational
Expires: September 2015

Z. Qiang
Robert Szabo
Ericsson

March 2, 2015

Elasticity VNF
draft-zu-nfvrg-elasticity-vnf-01.txt

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html

This Internet-Draft will expire on September 3, 2015.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Abstract

This draft is an analysis of Network Function Virtualization (NFV) applications based on the NFV architecture, use cases and requirements. The purpose of this analysis is to identify any NFV characteristics related issues. The analysis is focusing on elastic VNF with predicable performance, reliability and security. Only the issues which are unique to NFV are discussed in this document.

Table of Contents

1. Introduction

Network Functions Virtualization (NFV) is a network architecture concept that proposes using IT virtualization related technologies, to virtualize entire classes of network node functions into building blocks that may be connected, or chained, together to create communication services. NFV aims to transform the traditional operator architect networks by evolving standard IT virtualization technology to consolidate network equipment types onto industry standard high volume services, switches and storage, which could be located in a variety of NFV Infrastructure Point of Presences (NFVI PoPs) including Data Center (DC), network nodes and in end user premises. It is also indicated that an important part of controlling the NFV environment should be done through automation network management and orchestration.

This draft is an analysis of NFV applications based on the NFV architecture, use cases and requirements. The purpose of this analysis is to identify any NFV characteristics related issues. The analysis is focusing on elastic VNFs with predicable performance, reliability and security. Only the issues which are unique to NFV are discussed in this document. The intention is to identify what is missing, and what is needed to be addressed in terms of protocol / solution specifications which may be the potential work for IETF.

The reader is assumed to be familiar with the terminology as defined in the NFV document [nfv-tem].

2. Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC-2119 [RFC2119].

In this document, these words will appear with that interpretation only when in ALL CAPS. Lower case uses of these words are not to be interpreted as carrying RFC-2119 significance.

3. Terminology

This document uses the same terminology as found in the NFV end to end architecture [nfv-tem]:

Network Function Consumer: a Network Function Consumer (NFC) is the consumer of virtual network functions. It can be either an individual user, home user or the enterprise user.

NFV: network function virtualization. NFV technology uses the commodity servers to replace the dedicated hardware boxes for the network functions, for example, home gateway, enterprise access router, carrier grade NAT and etc. So as to improve the reusability, allow more vendors into the market, and reduce time to market. NFV architecture includes a NFV Control and Management Plane (orchestrator) to manage the virtual network functions and the infrastructure resources.

NF: A functional building block within an operator's network infrastructure, which has well-defined external interfaces and a well-defined functional behavior. Note that the totality of all network functions constitutes the entire network and services infrastructure of an operator/service provider. In practical terms, a Network Function is today often a network node or physical appliance.

Network Function Provider: a Network Function Provider (NFP) provides virtual network function software.

Network Service Provider (NSP): a company or organization that provides a network service on a commercial basis to third parties. A network service is a composition of network functions and defined by its functional and behavior specification. The NSP operates the NFV Control Plane.

NFV Infrastructure (NFVI): NFV Infrastructure indicates the computing, storage and network resources to implement the virtual network function. High performance acceleration platform is also part of it.

VNF: virtual network function, an implementation of an executable software program that constitutes the whole or a part of an NF that can be deployed on a virtualization infrastructure.

VM: virtual machines, a program and configuration of part of a host computer server. Note that the Virtual Machine inherits the

properties of its host computer server e.g. location, network interfaces.

NFV Control and Management Plane (NFVCMP): a NFV Control and Management Plane is operated by a NSP and orchestrates the NFV NFV Overview

4. Network Function Virtualization

4.1. NFV Requirements

There are many virtualization requirements described by NFV in [nfv-req]. The followings are highlights of a few NFV requirements which are related to this document:

- Portability: VNF portability is a reasonable generic virtualization requirement. It allows VNF mobility across different but standard multi-vendor environment. However, moving a VNF within the NFV framework with the Service Level Specification (SLA) requirements including performance, reliability and security could be a challenge.
- Performance: Virtualization adds additional processing overhead and increases the latency. For latency-sensitive VNFs, it is a big concern for NFV on how to achieve predictable low-latency performance.
- Elasticity: NFV elasticity requirement allows the VNF to be scaled within NFVI. Within the NFV framework, it is important to support VNF scaling with the SLA requirements including performance, reliability and security.
- Resiliency: NFV resiliency is a must requirement for NFV network, including both the control plane and data plane. Necessary mechanisms must be provided to improve the service availability and fault management.
- Security: The traditional telecom network functions are developed in dedicated hardware located in an isolated network. Security is provided by underlay network. When moving VNF into a DC network with shared Infrastructure, security becomes a big concern.
- Service Continuity: At VNF failure over, migration, mobility, and upgrading, service downtime may not be avoided. In NFV, service continuity must be supported which means the provided service must be restored at the VNF instance updated / replaced / recovered. This procedure includes the restoration of any ongoing data sessions. And it shall be transparent to the user of NFV service.

4.2. NFV Use Cases

Multiple use cases are described by NFV in [nfv-uc]. The followings are a highlight of the NFV use cases.

4.2.1. Network Function Virtualization Infrastructure

Network Function Virtualization Infrastructure as a Service (NFVIaaS), Virtual Network Function as a Service (VNFaaS) and Virtual Network Platform as a Service (VNPaaS) are the NFV use cases which describe how the telecom operators would like to build up their telecom cloud infrastructure using virtualization.

Network Function Virtualization Infrastructure (NFVI) is the totality of all hardware and software components which build up the environment in which VNFs are deployed. The NFVI can span across several locations. The network providing connectivity between these locations is regarded to be part of the NFVI.

NFVIaaS is a generic IaaS plus NaaS requirement which allows the telecom operator to build up a VNF cloud on top of their own DCs Infrastructure and any external DCs Infrastructure. This will allow a telecom operator to migrate some of its network functions into a 3rd party DC when it is needed. Furthermore, a larger telecom operator may have multiple DCs in different geography locations. The operator may want to setup multiple virtual data center (vDC), where each vDC may cross several of its physical DCs geography locations. Each vDC is defined for providing one specific function, e.g. Telco Cloud.

VNFaaS is more focusing on enterprise network which may have its own cloud infrastructure with some specific services / applications running. VNFaaS allows the enterprise to merge and/or extend its specific services / applications into a 3rd party commercial DC provided by a telecom operator. With this VNFaaS, the enterprise does not need to manage and control the NFVI or the VNF. However, NFV Performance & portability considerations will apply to deployments that strive to meet high performance and low latency considerations.
With VNPaaS, the mobile network traffic, including WiFi traffic, is routed based on the APN to a specific packet data service server over the mobile packet core network. Applications running at the packet data service server may be provided by the enterprise. And it is possible to have an interface to route the traffic into an enterprise network. But the infrastructure hosting the application is fully under controlled by the operator. However, the enterprise

has full admin control of the application and needs to apply all configurations on its own, potentially via a vDC like management interface with support of the hosting operator.

All the above use cases need solutions for the operator to share the infrastructure resources with 3rd parties. Therefore cross domain orchestration with access control is needed. Besides, the infrastructure resource management needs to provide a mechanism to isolate the traffic, not only based on the traffic type, but also from different operators and enterprises.

4.2.2. Telecom Network Functions Migration

Virtualization of telecom network functions, including Mobile Core Network functions, IMS functions, Mobile base station functions, Content Delivery Networks (CDN) functions, Home Environment functions, and Fixed Access Network functions, are described in the NFV use case document [nfv-uc]. In additional, VNF forwarding Graphs is another use case which describes how the user data packets are forwarded by traversing more than one operator service chain functions, such as DPI, Firewall, Content Filtering, before reaching the service server.
Migrate the telecom functions includes moving the control plane, data plane and service network into a cloud based network and using cloud based protocol to control the data plane. Service continuity, network security, service availability, resiliency in both control plane and data plane must be ensured at this migration.

5. Elasticity in a Distributed Cloud

Today the usage of personal devices, e.g. smartphones, for internet service traffic, telecom specific service access, and accessing the corporate network, is increased significantly. At the same time, telecom operators are under pressure to accommodate the increased service traffic in a fine-grained manner. Services provided by telecom network must be done in an environment of increased security, compliance, and auditing requirements, along with traffic load may be changed dramatically overtime. Providing self-service provisioning in telecom cloud requires elastic scaling of the VNF based on the dynamic service traffic load and resource management e.g. computing, storage, and networking.

The existing telecom network functions may not be cloud technologies ready yet. Most of the NFV functions are stateful and running on either specific hardware or a big VM. It is not designed to tolerate

any system failure in many VMs. The network functions are very difficult in term of configuration, scale updating, etc.

Re-engineering may be needed for virtualization enabling, e.g. software adaption for software and hardware decoupling. For cloud technologies readiness, telecom network functions need to be re-designed to run on small VMs with multiple instances which can provide higher application availability. Such VMs may be stateless in operations or may need to support state migration (e.g., OpenNF http://opennf.cs.wisc.edu/). With cloud ready network functions, applications' dynamic scaling can be achieved by adding more VMs into the service.

Virtualization provides the elasticity ability to scale up / down, scale out / in with guaranteed computational resources, security isolation and API access for provisioning it all, without any of the overhead of managing physical servers. However, there are still many optimizations which can be used to avoid the increasingly overhead.

5.1. NFV Infrastructure

Virtualized Network Function (VNF) is an implementation of a network function that can be deployed on Network Function Virtualization Infrastructure (NFVI).

For a large telecom operator, multiple NFVI Point of Presences (NFVI PoPs) may be created according to multiple physical data centers. As NFVI PoPs may be located in different geography locations, networking characteristics should be taken into account when selecting an NFVI PoP to host a VNF.

5.2. Elastic VNF

In many 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. In that case either stateless operation is needed, or the support of state information migration procedure is required, which will increase the complexities of the VNF implementation.

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 capacity may be limited if it only can be scaled within one NFVI PoP, e.g., within one DC in a geography location. As an NFVI which may be crossing multiple NFVI PoPs (or data center)s, it is possible

to scale an elastic VNF crossing different network zones if it is needed. At cross DC scaling, the result is that the new VNF instance may be placed at a remote cloud location. It is a must requirement to provide the same level of SLA including performance, reliability and security.

5.3. VNF Forwarding Graphs

In NFV network, a VNF Forwarding Graph (VNF FG) (an application) may consist of multiple VNFs, where each VNF may consist of multiple VNF instances. Normally the VNFs are working as such that the services provided by the VNFs may need to process the user data packets with several selected VNF instances before delivering it to its destination

For instance, when mobile users setup a PDN connection for IMS services, there are multiple network entities involved along the PDN connection, including eNB, Serving GW, PDN GW, P-CSCF, S-CSCF, etc. Another example is service function chaining, where a service chain is referring to one or more service processing functions in a specific order which are chained to provide a composite service.

In telecom cloud, a service session may traverse multiple stateful and stateless VNF functions of a VNF set. And with an NFVI consisting of multiple NFVI PoPs, it may be crossing multiple DCs. In such cloud, an incoming data packet may be processed by multi-VNF instance before delivering to the final destination. Therefore the east-west traffic (i.e. data traffic between VNFs within the DC) is much heavier comparing to the north-south traffic (i.e. data traffic in/out from the DC).
When placing VNF Forwarding Graphs, it is better spread the VNF components across many NFVI PoPs, which may give a better availability. However, multiple NFVI PoPs may also increases the network latency, which can be considerably big compared to latencies within a single NFVI PoP. Therefore, the whole VNF Forwarding Graph should be taken into account instead of a single VNF component during orchestration. Furthermore, during VNF scaling, dependencies (interconnection) with other service instances of the VNF Forwarding Graph shall also be considered.

When scaling, VNFs are not scaled only in relation to compute and storage PoPs. VNF instances may need to be grouped together according to the VNF FG and subjected to auto-scaling techniques to the entire group. The scaling policies, e.g., ratio between the

different VNFs, need to be applied on the VNF FG in aggregate to control the scaling process.

5.4. VNF scaling across multiple NFVI PoPs

Since in general, a VNF is part of a VNF Forwarding Graph (or a service function chain), 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 [unify1].

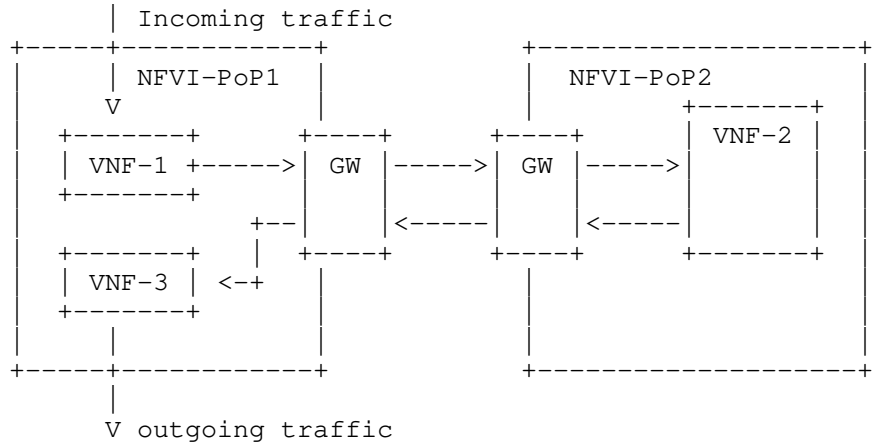


Figure 1 a data traffic flow traversing distributed VNF cloud

6. Elasticity with Predicable Performance

6.1. Predicable Performance

High performance with low-latency VNF is expected in the NFV framework. The NFVI metrics are related to any kind of metrics generated by the NFVI, including not only CPU load on a VM, CPU load

on a host, but also interrupt rate handled by the hypervisor or network latency/packet loss.

Virtualization adds additional overhead which impacts the performance. This additional extra distortion shall be avoided or, at least, minimized. It is a big concern for NFV on how to achieve predictable and low-latency performance, not only at placing a VNF into the DC, but also at VNF scaling.

Operator may wish to run standard test and use the result to provide KPIs of the VNF. A significant part of a VNF vendor's performance guarantees will depend on the choice of the virtualization technology.
Network latency may not be at the same level if the physical connections between the servers are various. Furthermore, geography location of the physical servers also increases the network latency. When placing or moving a VNF, the location of other VNFs of the same VNF set shall be considered to avoid network latency issue. For instance, VNF-a, VNF-b, and VND-c are grouped as one VNF set. When moving VNF-b into a new location, the network connection between the new location and the existing location may be a concern. As the traffic may traverse from VNF-a to VNF-b, then VND-c, moving the VNF-b into a new location may create Ping-Pong type of traffic, which the network latency may be doubled. The best choice would be to move the whole VNF set into the new location instead of only one VNF.

6.2. Hardware virtualization features

Virtualization layer adds minimal overhead and delivers a predictable performance between a minimum and maximum threshold for latency and jitter which are far more important. Light weight virtualization, e.g. container or bare metal, may be considered for performance sensitive VNF applications. In additional, hardware virtualization features (e.g. SR-IOV) are important to be supported in order to provide some performance improvement. Many VNF requires direct access to the device hardware so that they can offload functionality with throughput rates of millions of packets a second. Another alternative, which may be more attractive for latency-sensitive applications, is using and non-hypervisor virtualization, including bare metal and Linux container.
Optimization to drive high-throughput network workloads associated with such functions as traffic filtering, NATing and firewalling. Avoiding performance bottleneck, the virtualization layer shall have a suitably-architected I/O stack.

6.3. Network Overlay

Network overlay adds additional overhead when forwarding the data packets. Reference [vxlan-p] is a VXLAN performance testing report which indicates the overlay performance is a concern. Avoiding overlay connections may be one option which is more attractive for latency-sensitive applications.

Furthermore, additional network latency may be added when traversing the cross-DC overlay connections. To avoid any additional network latency, all the functions of a VNF set may be placed in the same low-latency network zone, e.g. same host or same DC. However, when the capacity limitation the network zone is reached, scaling-out one VNF into another network zone may be needed. In this case, as the service session has to traverse the same path, the Ping-Pong traffic between the network zones cannot be avoided. Depends on the network overlay technologies used for the cross network zone connection, the overhead network latency can be various. In another words, the network performance may become unpredictable.

7. Elasticity with Reliability

NFV resiliency is a must requirement for NFV network, including both the control plane and data plane. Necessary mechanisms must be provided to improve the service availability and fault management.

With virtualization, the use of VNFs can pose additional challenges on the reliability of the provided services. For a VNF instance, it typically would not have built-in reliability mechanisms on its host (i.e., a general purpose server). Instead, there are more factors of risk such as software failure at various levels including hypervisors and virtual machines, hardware failure, and instance migration that may make a VNF instance unreliable. Even for cloud ready NFV applications, a HA may still be needed as the storage, load balancer may be failure. Service restoration solution is still needed.

One alternative to improve the VNF resiliency is to take snapshot of the VM periodically. At VNF failure, the network can restore the VM at same or different host using the stored snapshot. However, there is a downtime of the provided service due to the snapshot recovering. And the downtime is much longer than the expected value which could be tolerated by NFV. NFV has a completely different level of reliability requirements, e.g. recovering time, comparing to enterprise cloud applications.

To improve the network function resiliency, some kind high availability (HA) solutions may be needed for NFV network, which has the potential to minimize the service downtime at failure. However, in most of the telecom use cases, there are application level restoration procedures available which makes the high availability solution less important.

The VNF reliability can be achieved by eliminating any single points of failure by creating a redundancy of resources, normally, including enough excess capacity in the design to compensate for the performance decline and even failure of individual resources; that is, a group of VNF instances providing the same function works as a network function cluster or pool, which provides protection (e.g. failover) for the applications and therefore an increased availability.

8. Elasticity with Security

TDB

9. Security Considerations

This is a discussion paper which provides inputs for NFV related discussions and in itself does not introduce any new security concerns.

10. IANA Considerations

No actions are required from IANA for this informational document.

11. References

11.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2234] Crocker, D. and Overell, P.(Editors), "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, Internet Mail Consortium and Demon Internet Ltd., November 1997.

11.2. Informative References

[nfv-arch] Network Functions Virtualization Infrastructure Architecture Overview; GS NFV INF 001.

[nfv-rel] Network Function Virtualization (NFV) Resiliency Requirements; ETSI GS NFV-REL 001.

[nfv-uc] Network Function Virtualization (NFV) Use Cases; ETSI GS NFV 001

[nfv-req] Network Function Virtualization (NFV) Virtualization Requirements; ETSI GS NFV 004

[nfv-sec] Network Function Virtualization (NFV) NFV Security Problem Statement; ETSI NFV-SEC 001

[nfv-tem] Network Function Virtualization (NFV) Terminology for Main Concepts in NFV; ETSI GS NFV 003

[vxlan-p] Problem Statement for VxLAN Performance Test, draft-liu-nvo3-ps-vxlan-perfomance, (working in progress)

[unify1] Szabo, R., Csaszar, A., Pentikousis, K., Kind, M., and D. Daino, "Unifying Carrier and Cloud Networks: Problem Statement and Challenges", draft-unify-nfvrg-challenges-00 (work in progress), October 2014.

12. Acknowledgments

Many people have contributed to the development of this document and many more will probably do so before we are done with it. While we cannot thank all contributors, some have played an especially prominent role. The following have provided essential input: Suresh Krishnan.

Authors' Addresses

Zu Qiang
Ericsson
8400, boul. Decarie
Ville Mont-Royal, QC,
Canada

Email: Zu.Qiang@Ericsson.com

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

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