NFVRG   R. Szabo
Internet-Draft   A. Csaszar
Intended status: Informational   Ericsson
Expires: January 9, 2017   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

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.

```
                       fr2  +---+ fr3
                       +->o-|NF2|-o-+
                       |   4 +---+ 5 |
               +---+   |             V +---+
          1-->o-|NF1|-o----------->o-|NF3|-o-->8
               2 +---+ 3      fr1     6 +---+ 7
```
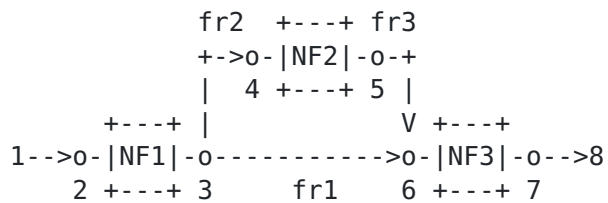
                       Figure 1: Service graph

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

   o corresponding ports (NF1:{2,3}; NF2:{4,5}; NF3:{6,7})

   o service access points {1,8} corresponding to infrastructure
      resources,

   o 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.

```
                          +---+
                      +--|SW3|--+
                      |   +---+   |
         +---+        |           |        +---+
         1  |PoP|     +---+       +---+     |PoP|  8
         o--|DC1|----|SW2|------|SW4 |---|DC2|--o
            +---+     +---+       +---+     +---+

         [---SP1---][--------SP2-------][---SP3----]
```
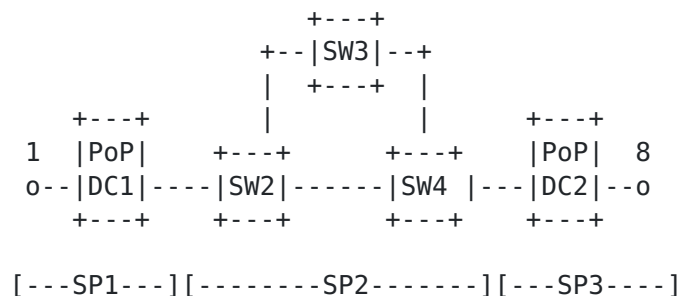
                    Figure 2: Infrastructure resources

```
              +----------+
              |  +----+  |PoP DC (== NFVI PoP)
              |  | CN |  |
              |  +----+  |
              |   | |    |
              |  +----+  |
           o-+--| SW |--+-o
              |  +----+  |
              +----------+
```
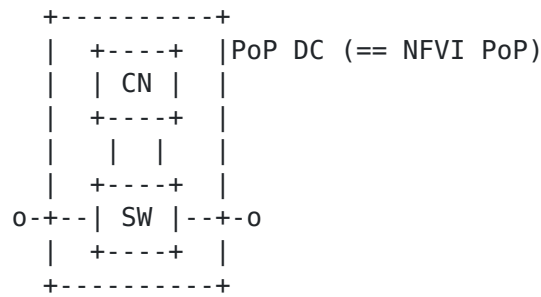
           Figure 3: A virtualized Point of Presence (PoP) with software
                     resources (Compute Node - CN)

```
             +----------+
             |  +----+  | UN
             |  | CN |  |
          o-+--+----+--+-o
             |  | SW |  |
             |  +----+  |
             +----------+
```

          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
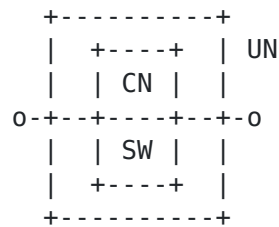Node (UN), an innovative element that integrates on the same platform
both compute and networking components and that could be used in the
infrastructure as an alternative to elements depicted in Figure 2 for
what concerns network and/or compute resources.

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.

```
                      fr2  +---+  fr3
                      +->o-|NF2|-o-+
                      |  4 +---+ 5 |
             +---+ |              V +---+
        1-->o-|NF1|-o----------->o-|NF3|-o-->8
           2 +---+ 3      fr1    6 +---+ 7


                           ||
                           ||
        +--------+         \/          SP0
        |   NF   |    +--------------------+
        |Resource|==>|Resource Orchestrator|==> MAPPING
        | Descr. |   |        (RO)         |
        +--------+   +--------------------+
                           /\
                           ||
                           ||


                         +---+
                      +--|SW3|--+
                      |  +---+  |
        +---+         |         |       +---+
        1 |PoP|      +---+     +---+     |PoP|  8
        o--|DC1|-----|SW2|-----|SW4|----|DC2|--o
          +---+      +---+     +---+     +---+

        [---SP1---][--------SP2-------][---SP3----]
        [------------------SP0-------------------]
```
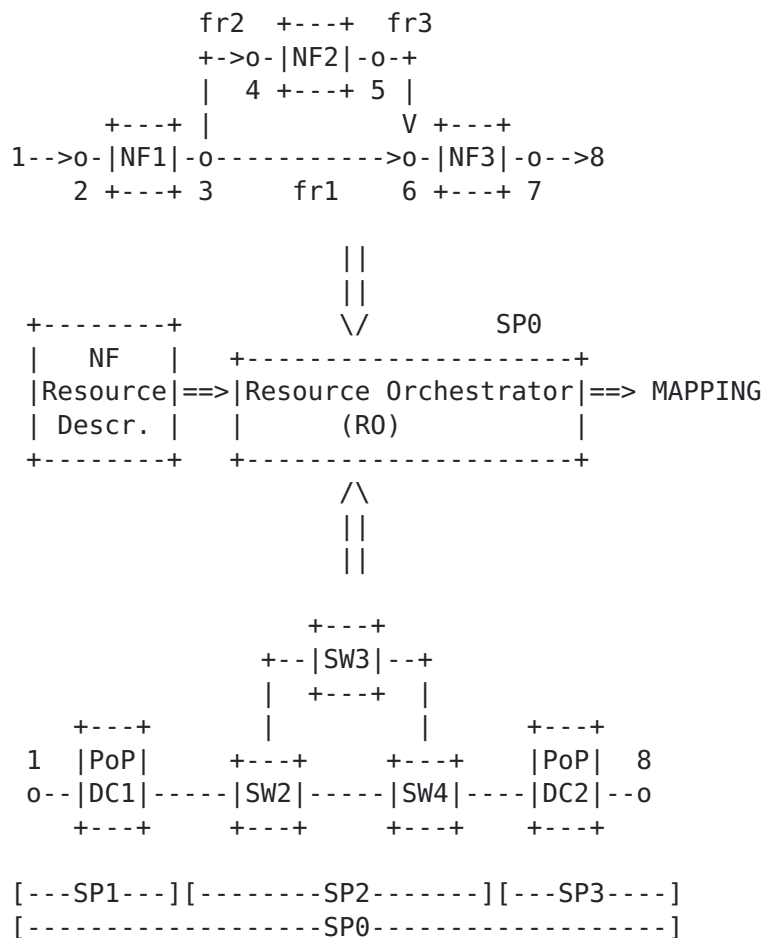
       Figure 5: Resource Orchestrator: information base, inputs and output

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.

```
                              +---------+
                              |Orchestr.|
                              |   SP0   |
                           ___+---------+___
                          /        |        \
                         /         V Ic2      \
                        |     +---------+      |
                 Ic1 V        |SDN Ctrl |     V  Ic3
               +---------+    |   SP2   |    +---------+
               |Comp Ctrl|    +---------+    |Comp Ctrl|
               |  SP1    |      /  |  \      |  SP3    |
               +---------+    +---  V  ----+ +---------+
                    |          |  +----+   |      |
                    |          |  |SW3 |   |      |
                    V          |  +----+   |      V
                 +----+      V /       \ V     +----+
               1 |PoP |    +----+      +----+  |PoP |  8
               o--|DC1 |----|SW2 |------|SW4 |----|DC2 |--o
                 +----+    +----+      +----+   +----+

            [----SP1---][---------SP2--------][---SP3----]
            [--------------------SP0--------------------]
```
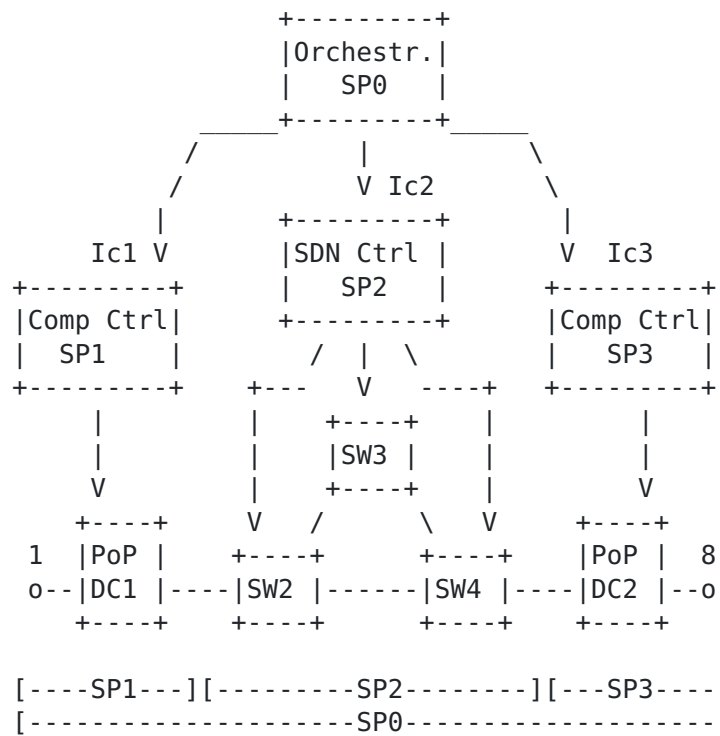
   Figure 6: The RO Control Plane view. Control plane interfaces are
   indicated with (line) arrows. Data plane connections are indicated
                          with simple lines.

   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.

```
                           NBI
                            |
                      +---------+
                      |Comp Ctrl|
                      +---------+
                  +----+      |
            IcA V          | IcB:to CNs
         +---------+         V
         |SDN Ctrl |    |             |  ext port
         +---------+  +---+        +---+
            to|SW       |SW |        |SW |
              +->      ,+--++.._   _+-+-+
              V     ,-"    _|,,`."""-..+
               _,,,--""" |      `.  |""-.._
            +---+      +--++       `+-+-+    ""+---+
            |SW |      |SW |        |SW |       |SW |
            +---+    ,'+---+      ,'+---+     ,'+---+
            |   | ,-" |   |   | ,-" |   |   | ,-" |   |
         +--+ +--+  +--+ +--+  +--+ +--+   +--+ +--+
         |CN| |CN|  |CN| |CN|  |CN| |CN|   |CN| |CN|
         +--+ +--+  +--+ +--+  +--+ +--+   +--+ +--+
```
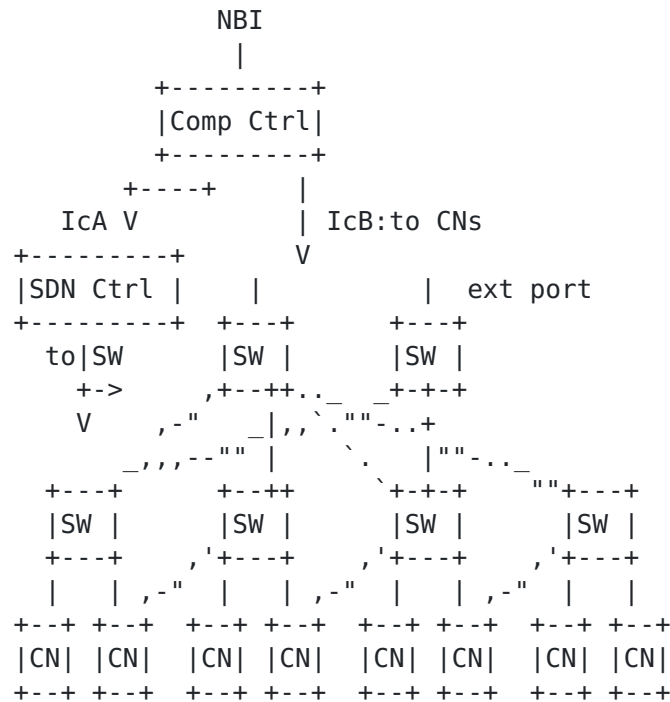
            Figure 7: PoP DC Network with Compute Nodes (CN)

   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.

```
                      +-+   +-+ +-+   +-+
                      |V|   |V| |V|   |V|
                      |N|   |N| |N|   |N|
                      |F|   |F| |F|   |F|
                      +-+   +-+ +-+   +-+
                      |   /   /       |
                      +---+ +---+ +---+
                      |LSI| |LSI| |LSI|
                      +---+ +---+ +---+
                       |  /        |
                      +---+      +---+
                      |NIC|      |NIC|
                      +---+      +---+
                       |           |
```
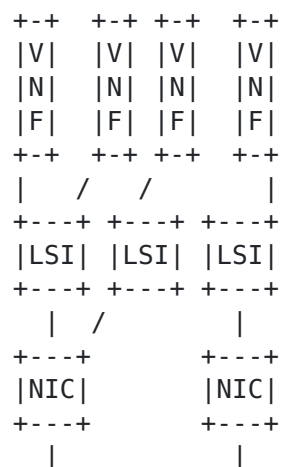
            Figure 8: Compute Node with internal switching resource

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,
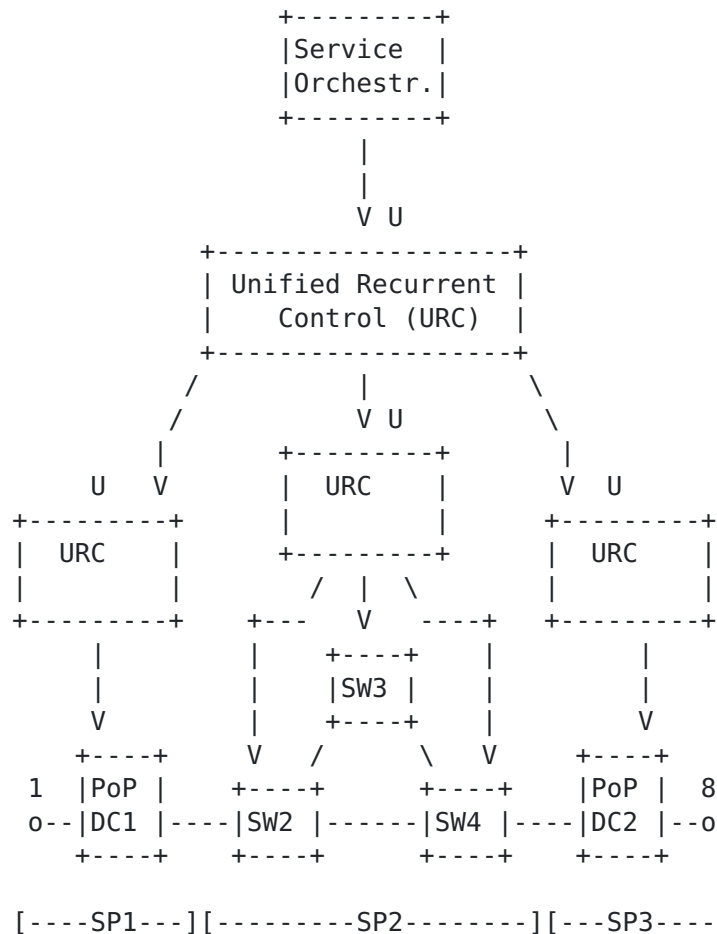storage and network provisioning as depicted in Figure 9.

```
                            +---------+
                            |Service  |
                            |Orchestr.|
                            +---------+
                                 |
                                 |
                                 V U
                     +-------------------+
                     | Unified Recurrent |
                     |    Control (URC)  |
                     +-------------------+
                      /         |        \
                     /          V U        \
                    |      +---------+       |
            U     V       |   URC   |      V   U
        +---------+       |         |      +---------+
        |   URC   |       +---------+      |   URC   |
        |         |        / | \           |         |
        +---------+   +---  V  ----+       +---------+
             |         |   +----+   |           |
             |         |   |SW3 |   |           |
             V         |   +----+   |           V
           +----+      V  /      \  V         +----+
        1  |PoP |    +----+      +----+       |PoP |  8
        o--|DC1 |----|SW2 |------|SW4 |----|DC2 |--o
           +----+    +----+      +----+       +----+

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

   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
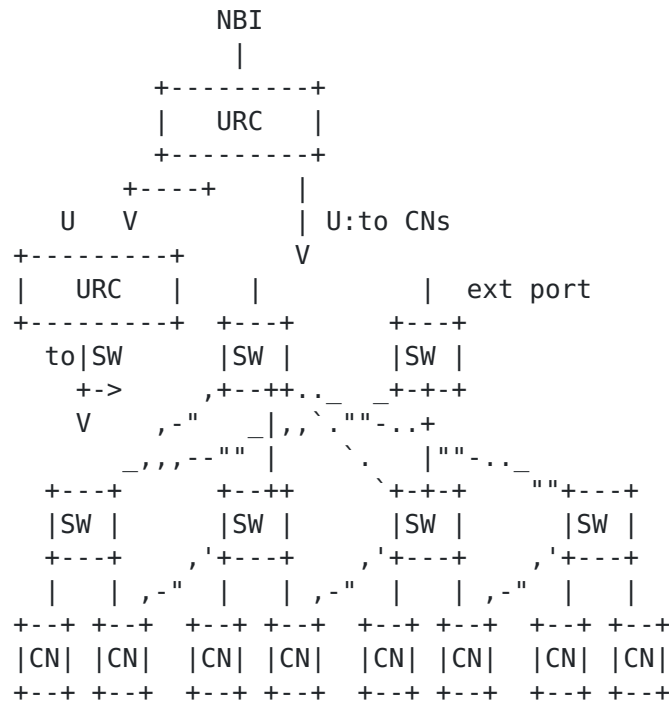Nodes view changes as reported in Figure 10.

```
                           NBI
                            |
                      +---------+
                      |  URC    |
                      +---------+
                   +----+       |
                   U    V       | U:to CNs
                +---------+      V
                |  URC    |   |              | ext port
                +---------+  +---+       +---+
                  to|SW       |SW |        |SW |
                   +->     ,+--++.._   _+-+-+
                    V    ,-"     _|,,`."""-..+
                     _,,,--""  |     `.    |"""-.._
                   +---+     +--++     `+-+-+     ""+---+
                   |SW |      |SW |        |SW |       |SW |
                   +---+    ,'+---+     ,'+---+    ,'+---+
                   |   | ,-" |    | ,-" |    | ,-" |    |
                 +--+ +--+  +--+ +--+  +--+ +--+   +--+ +--+
                 |CN| |CN|  |CN| |CN|  |CN| |CN|   |CN| |CN|
                 +--+ +--+  +--+ +--+  +--+ +--+   +--+ +--+
```

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