NFV Research Group   N. Figueira
Internet-Draft   Brocade
Intended status: Informational   R. Krishnan
Expires: February 21, 2016   Dell
                                                              D. Lopez
                                                            Telefonica
                                                             S. Wright
                                                                  AT&T
                                                       D. Krishnaswamy
                                                                   IBM
                                                       August 20, 2015

        Policy Architecture and Framework for NFV Infrastructures
                draft-irtf-nfvrg-nfv-policy-arch-01

Abstract

   A policy architecture and framework is discussed to support NFV
   environments, where policies are used to enforce business rules and
   to specify resource constraints in a number of subsystems. This
   document approaches the policy framework and architecture from the
   perspective of overall orchestration requirements for services
   involving multiple subsystems. The framework extends beyond common
   orchestration constraints across compute, network, and storage
   subsystems to include energy conservation. This document also
   analyses policy scope, global versus local policies, static versus
   dynamic versus autonomic policies, policy actions and translations,
   policy conflict detection and resolution, interactions among policies
   engines, and a hierarchical policy architecture/framework to address
   the demanding and growing requirements of NFV environments. These
   findings may also be applicable to cloud infrastructures in general.

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 May 2015.

Copyright Notice

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**

   This document discusses the policy architecture and framework to
   support Network Function Virtualization (NFV) [3] infrastructures. In
   these environments, policies are used to enforce business rules and
   to specify resource constraints, e.g., energy constraints, in a
   number of subsystems, e.g., compute, storage, network, and etc., and
   across subsystems. These subsystems correspond to the different
   "infrastructure domains" identified by the NFV ISG Infrastructure
   Working Group [6][7].

   The current work in the area of policy for NFV is mostly considered
   in the framework of general cloud services, and typically focused on
   individual subsystems and addressing very specific use cases or
   environments. For example, [3] addresses network subsystem policy for
   network virtualization, [15] and [16] are open source projects in the
   area of network policy as part of the OpenDaylight [17] software
   defined networking (SDN) controller framework, [14] specifies an
   information model for network policy, [9] focuses on placement and
   migration policies for distributed virtual computing, [19] is an open
   source project proposal in OpenStack [18] to address policy for
   general cloud environments.

   This document approaches policy, policy framework, and policy
   architecture for NFV services from the perspective of overall
   orchestration requirements for services involving multiple
   subsystems, and can be applied to the general case of any cloud-based
   service. The analysis extends beyond common orchestration constraints
   across compute, network, and storage subsystems to also include
   energy conservation constraints applicable to NFV and other
   environments. The analysis in this document also extends beyond a
   single virtual Point of Presence (vPoP) or administrative domain to
   include multiple data centers and networks forming hierarchical
   domain architectures [8]. The focus of this document is not general
   policy theory, which has already been intensively studied and
   documented on numerous publications over the past 10 to 15 years (see
   [14], [22], [13], [23], and [1] to name a few). This document's
   purpose is to discuss and document a policy architecture that uses
   known policy concepts and theories to address the unique requirements
   of NFV services including multiple vPoPs and networks forming
   hierarchical domain architectures [8].

   With the above goals, this document analyses policy scope, global
   versus local policies, static versus dynamic versus autonomic
   policies, policy actions and translations of actions, policy conflict
   detection and resolution (which can be relevant to resource
   management in service chains [12]), the interactions among policies
   engines from the different vPoPs and network subsystems, and a

hierarchical policy architecture/framework to address the demanding
and growing requirements of NFV environments. These findings may also
be applicable to cloud infrastructures in general.

[2](#). **Policy Intent Statement versus Subsystem Actions and Configurations**

Policies define which states of deployment are in compliance, and, by
logic negation, which ones are not. The compliance statement in a
policy may define specific actions, e.g., "a given customer is [not
allowed to deploy VNF X]", where VNF refers to a Virtual Network
Function, or quasi-specific actions, e.g., "a given customer [must be
given platinum treatment]." Quasi-specific actions differ from the
specific ones in that the former requires an additional level of
translation or interpretation, which will depend on the subsystems
where the policy is being evaluated, while the latter does not
require further translation or interpretation.

In the previous examples, "VNF X" defines a specific VNF type, i.e.,
"X" in this case, while "platinum treatment" could be translated to
an appropriate resource type depending on the subsystem. For example,
in the compute subsystem this could be translated to servers of a
defined minimum performance specification, while in the network
subsystem this could be translated to a specific Quality of Service
(QoS) level treatment.

The actions defined in a policy may be translated to subsystem
configurations. For example, when "platinum treatment" is translated
to a specific QoS level treatment in a networking subsystem, one of
the outcomes (there can be multiple ones) of the policy could be the
configuration of network elements (physical or virtual) to mark that
customer's traffic to a certain DSCP (DiffServ Code Point) level
(Figure 1). Some may refer to the QoS configuration above as a policy
in itself, e.g., [22], [23], [2], and [13]. In this document, such
domain configurations are called policy enforcement technologies to
set them apart from the actual policy intent, e.g., "a given customer
must be given platinum treatment" as in the above example.

Describing intent using a high-level policy language instead of
directly describing configuration details allows for the decoupling
of the desired intent from the actual configurations, which are
subsystem dependent, as shown in the previous example (Figure 1). The
translation of a policy into appropriate subsystem configurations
requires additional information that is usually subsystem and
technology dependent. Therefore, policies should not be written in
terms of policy enforcement technologies. Policies should be
translated at the subsystems using the appropriate policy provides a
few examples where the policy "a given customer must be given

platinum treatment" is translated to appropriate configurations at the respective subsystems.

The above may sound like a discussion about "declarative" versus "imperative" policies. We are actually postulating that "imperative policy" is just a derived subsystem configuration using an appropriate policy enforcement technology to support an actually intended policy.

```
+-------------------------------------------------------------------+
|   Policy: "a given customer must be given Platinum treatment"   |
+-------------------------------------------------------------------+
        ^                   ^                   ^                   ^
        |                   |                   |                   |
        V                   V                   V                   V
+-------------+   +-------------+   +-------------+   +-------------+
|Compute      |   |Network      |   |Storage      |   |Whatever     |
|Subsystem    |   |Subsystem    |   |Subsystem    |   |Subsystem    |
|             |   |             |   |             |   |             |
|Policy       |   |Policy       |   |Policy       |   |Policy       |
|translation: |   |translation: |   |translation: |   |translation: |
|             |   |             |   |             |   |             |
|Install      |   |Give customer|   |Give customer|   | ...         |
|customer VMs |   |the best QoS,|   |the fastest  |   |             |
|on servers   |   |which        |   |SSD storage. |   |             |
|with 3GHz    |   |translates   |   |             |   |             |
|16-core Xeon |   |here to set  |   |             |   |             |
|processors,  |   |DHCP to xx,   |   |             |   |             |
|and etc.     |   |and etc.     |   |             |   |             |
+-------------+   +-------------+   +-------------+   +-------------+
```

Figure 1: Example of Subsystem Translations of Policy Actions

[3]. Global vs Local Policies

Some policies may be subsystem specific in scope, while others may have broader scope and interact with multiple subsystems. For example, a policy constraining certain customer types (or specific customers) to only use certain server types for VNF or Virtual Machine (VM) deployment would be within the scope of the compute subsystem. A policy dictating that a given customer type (or specific customers) must be given "platinum treatment" could have different implications on different subsystems. As shown in Figure 1, that "platinum treatment" could be translated to servers of a given performance specification in a compute subsystem and storage of a given performance specification in a storage subsystem.

Policies with broader scope, or global policies, would be defined
outside affected subsystems and enforced by a global policy engine
(Figure 2), while subsystem-specific policies or local policies,
would be defined and enforced at the local policy engines of the
respective subsystems.

Examples of sub-system policies can include thresholds for
utilization of sub-system resources, affinity/anti-affinity
constraints with regard to utilization or mapping of sub-system
resources for specific tasks, network services, or workloads, or
monitoring constraints regarding under-utilization or over-
utilization of sub-system resources.

```
+--------------------------------------------------------------------+
|          +-----------------------------------------------+         |
|          |              Global Policy Engine         |         |
|          +-----------------------------------------------+         |
|                                                                    |
|          +-----------------------------------------------+         |
|          |              Global Policies               |         |
|          +-----------------------------------------------+         |
+--------------------------------------------------------------------+
        ^                   ^                   ^                   ^
        |                   |                   |                   |
        V                   V                   V                   V
+-------------+   +-------------+   +-------------+   +-------------+
|Compute      |   |Network      |   |Storage      |   |Whatever     |
|Subsystem    |   |Subsystem    |   |Subsystem    |   |Subsystem    |
|             |   |             |   |             |   |             |
|Local Policy |   |Local Policy |   |Local Policy |   |Local Policy |
|Engine       |   |Engine       |   |Engine       |   |Engine       |
|             |   |             |   |             |   |             |
|Local        |   |Local        |   |Local        |   |Local        |
|Policies:    |   |Policies     |   |Policies     |   |Policies     |
| P0, P1,     |   | P0, P1,     |   | P0, P1,     |   | P0, P1,     |
|             |   |             |   |             |   |             |
+-------------+   +-------------+   +-------------+   +-------------+
```
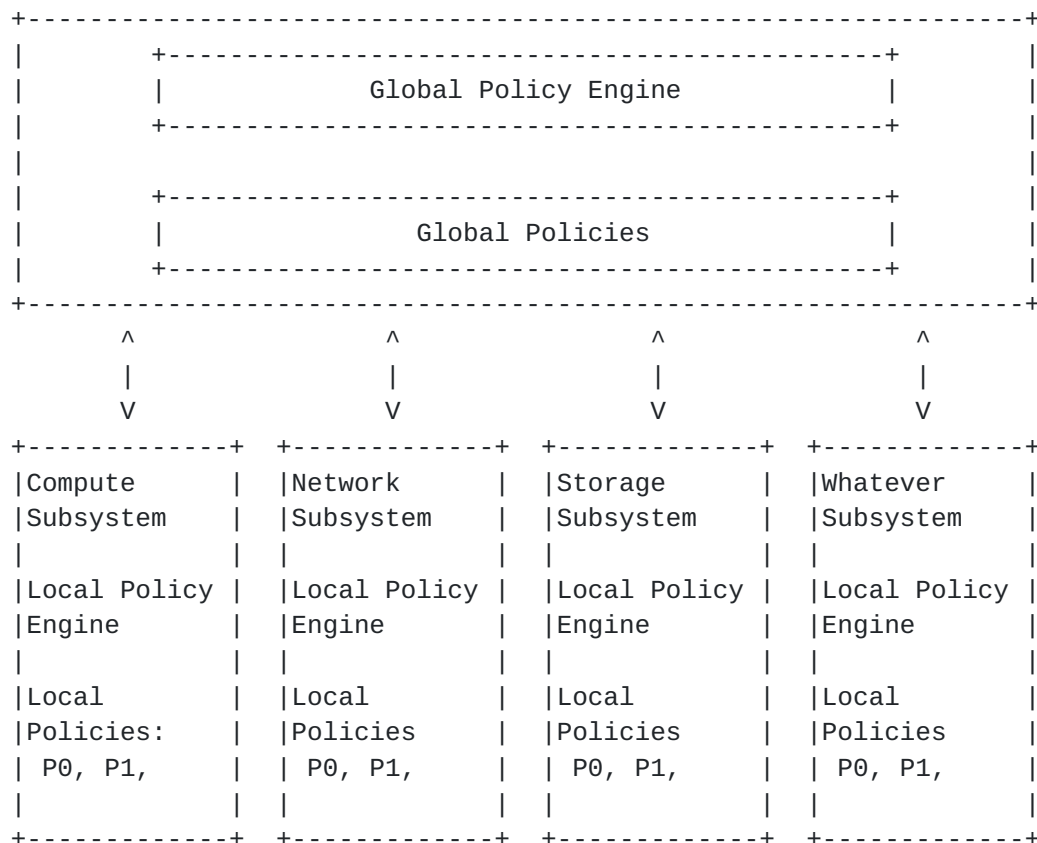
Figure 2: Global versus Local Policy Engines

4. Static vs Dynamic vs Autonomic Policies

Policies can be defined based on a diverse set of constraints and
sources, e.g., an operator's energy cost reduction policy based on
the time-varying energy rates imposed by a utility company supplying
energy to a given region or an operator's "green" policy based on the

operator's green operational requirements, which may drive a
reduction in energy consumption despite of potentially increased
energy costs.

While "static" policies can be imposed based on past learned behavior
in the systems, "dynamic" policies can be define that would override
"static" policies due to dynamically varying constraints. For
example, if a system needs to provided significantly additional
scaling of users in a given geographic area due to a sporting or a
concert event at that location, then a dynamic policy can be
superimposed to temporarily override a static policy to support
additional users at that location for a certain period of time.
Alternatively, if energy costs significantly rise on a particular
day, then an energy cost threshold could be dynamically raised to
avoid policy violation on that day.

Support for autonomic policies may also be required, such as an auto-
scaling policy that allows a resource (compute/storage/network/energy
resource) to be scaled up or down as needed. For example, a policy
specifying that a VNF can be scaled up or down to accommodate traffic
needs.

[5]. Hierarchical Policy Framework

So far, we have referenced compute, network, and storage as
subsystems examples. However, the following subsystems may also
support policy engines and subsystem specific policies:

    - SDN Controllers, e.g., OpenDaylight [17].
    - OpenStack [18] components such as, Neutron, Cinder, Nova, and
    etc.
    - Directories, e.g., LDAP, ActiveDirectory, and etc.
    - Applications in general, e.g., standalone or on top of
    OpenDaylight or OpenStack.
    - Physical and virtual network elements, e.g., routers, firewalls,
    application delivery controllers (ADCs), and etc.
    - Energy subsystems, e.g., OpenStack Neat [20].

Therefore, a policy framework may involve a multitude of subsystems.
Subsystems may include other lower level subsystems, e.g., Neutron
[21] would be a lower level subsystem in the OpenStack subsystem. In
other words, the policy framework is hierarchical in nature, where
the policy engine of a subsystem may be viewed as a higher level
policy engine by lower level subsystems. In fact, the global policy
engine in Figure 2 could be the policy engine of a Data Center
subsystem and multiple Data Center subsystems could be grouped in a
region containing a region global policy engine. In addition, one

could define regions inside regions, hierarchically, as shown in
Figure 3.

Metro and wide-area network (WAN) used to interconnect data centers
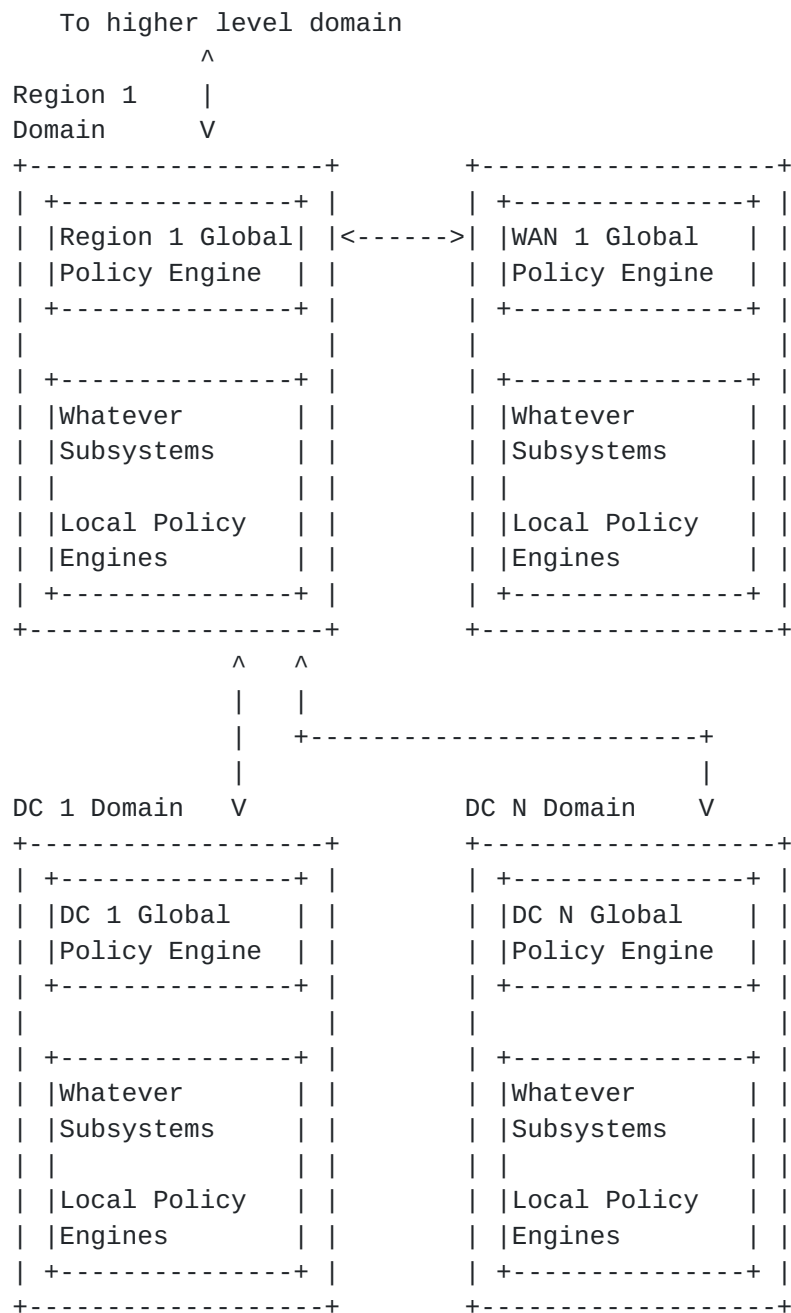would also be independent subsystems with their own policy engines.

```
          To higher level domain
                    ^
        Region 1    |
        Domain      V
        +------------------+        +------------------+
        | +--------------+ |        | +--------------+ |
        | |Region 1 Global| |<------>| |WAN 1 Global  | |
        | |Policy Engine  | |        | |Policy Engine | |
        | +--------------+ |        | +--------------+ |
        |                  |        |                  |
        | +--------------+ |        | +--------------+ |
        | |Whatever      | |        | |Whatever      | |
        | |Subsystems    | |        | |Subsystems    | |
        | | |            | |        | | |            | |
        | |Local Policy  | |        | |Local Policy  | |
        | |Engines       | |        | |Engines       | |
        | +--------------+ |        | +--------------+ |
        +------------------+        +------------------+
                  ^     ^
                  |     |
                  |     +------------------------+
                  |                              |
        DC 1 Domain    V            DC N Domain     V
        +------------------+        +------------------+
        | +--------------+ |        | +--------------+ |
        | |DC 1 Global   | |        | |DC N Global   | |
        | |Policy Engine | |        | |Policy Engine | |
        | +--------------+ |        | +--------------+ |
        |                  |        |                  |
        | +--------------+ |        | +--------------+ |
        | |Whatever      | |        | |Whatever      | |
        | |Subsystems    | |        | |Subsystems    | |
        | | |            | |        | | |            | |
        | |Local Policy  | |        | |Local Policy  | |
        | |Engines       | |        | |Engines       | |
        | +--------------+ |        | +--------------+ |
        +------------------+        +------------------+
```

             Figure 3: A Hierarchical Policy Framework

[6](#). **Policy Conflicts and Resolution**

Policies should be stored in databases accessible by the policy engines. For example, the local policies defined for the Compute subsystem in Figure 2 would be stored in a database accessible by the local policy engine in that subsystem.

As a new policy is added to a subsystem, the subsystem's policy engine should perform conflict checks. For example, a simple conflict would be created if a new policy states that "customer A must not be allowed to use VNF X", while an already existing policy states that "customer A is allowed to use VNF X". In this case, the conflict should be detected and an appropriate policy conflict resolution mechanism should be initiated.

The nature of the policy conflict resolution mechanism would depend on how the new policy is being entered into the database. If an administrator is manually attempting to enter that policy, the conflict resolution could entail a warning message and rejection of the new policy. The administrator would then decide whether or not to replace the existing policy with the new one.

When policies are batched for later inclusion in the database, the administrator should run a preemptive conflict resolution check on those policies before committing to include them in the database at a future time. However, running a preemptive conflict resolution check does not guarantee that there will be no conflicts at the time the batched policies are actually included in the database, since other policies could have been added in the interim that cause conflicts with those batched policies.

To avoid conflicts between batched policies waiting for later inclusion in the database and new policies being immediately added to the database, one could run a preemptive conflict resolution check against database policies and also batched policies every time new policies are added to the database. However, this may not be sufficient in case of separate administrative domains. A region administration could define batched polices to be pushed to the Compute subsystem of a Data Center at a later time. However, the Compute subsystem may be a separate administrative domain from that of the region administrative domain. In this case, the Compute subsystem may not be allowed to run preemptive policy conflict checks against the batched policies defined at the region administrative domain. Thus, there is a need for a reactive policy conflict resolution mechanism besides preemptive techniques.

The above discussions implicitly assumed that policies are individually evaluated for conflicts and individually committed

without regard to other policies. However, a set of policies could be labeled as part of a same "Commit Group", where the whole set of policies in the Commit Group must be committed for a desired result to be obtained. In this case, the conflict resolution mechanism would need to verify that none of the policies in the Commit Group conflicts with currently committed policies before the Commit Group is added (in other words, committed) to the policy database.

The Commit Group conflict detection mechanism and subsequent addition to the database should be implemented as an atomic process, i.e., no changes to the policy database should be allowed by other processes until either the whole Commit Group is checked and committed or a conflict is detected and the process stopped, to avoid multiple writers issues.

The above described atomic Commit Group conflict detection and policy commit mechanism would eliminate the need for Commit Group rollback. A rollback could be required if policies in a Commit Group were to be checked for conflicts and committed one by one, since the detection of a subsequent policy conflict in the Commit Group would require the rollback of previously committed policies in that group.

6.1. Soft vs Hard Policy Constraints

Policies at any level in the policy hierarchy can be either soft or hard. A soft policy imposes a soft constraint in a system that can be violated without causing any catastrophic failure in the system. A hard policy imposes a hard constraint in the system that must not be violated. An example of a soft constraint is the degree of underutilization (for example, a 40% utilization threshold could be used as a soft constraint) of compute servers with regard to CPU utilization. In such a case, when this soft constraint is violated, the system continues to function, although it may consume more energy (due to a non-linear dependence of the energy utilization as a function of the CPU utilization) compared to a task allocation across multiple servers after workload consolidation is performed. Alternatively, a soft constraint could be violated if the network bandwidth exceeds a certain fraction (say 80%) of the available bandwidth, the energy utilization exceeds a certain value, or the memory utilization falls below a certain value (say 30%). An example of a hard constraint could be to disallow a desired mean CPU utilization across allocated workloads in a compute subsystem to exceed a certain high threshold (such as 90%), or to disallow the number of CPUs allocated for a set of workloads to exceed a certain value, or to disallow the network bandwidth across workloads to exceed a certain value (say 98% of the maximum available bandwidth).

When considering policy conflicts, violation of policies across hard

policy constraints is undesirable, and must be avoided. A conflict resolution could be possible by relaxing one or more of the hard constraints to an extent that is mutually satisfactory to the imposers of the policies. Alternatively, as discussed earlier, a new hard policy that conflicts with existing policies is not admitted into the system. Violation of policies across soft policy constraints or between one or more hard policy constraints and one or more soft policy constraints can be allowed, such that one or more soft policy constraints are violated without hard constraints being violated. Despite soft constraints being violated, it is desirable to have a region of operating conditions that would allow the system to operate. For admission of new policy constraints, whether hard or soft, one should ensure that the overall system has a feasible region for operation given the existing constraints, and the new constraints under consideration. When such a feasible region is not possible, one can consider relaxing one or more of the existing or new constraints to allow such policies to be admitted.

7. Policy Pub/Sub Bus

In the previous section, we considered policy conflicts within a same level subsystem. For example, new local policies added to the Compute subsystem conflicting with existing local policies at that subsystem. However, more subtle conflicts are possible between global and local policies.

A global policy may conflict with subsystems' local policies. Consider the following Compute subsystem local policy: "Platinum treatment must be provided using server of type A."

The addition of the Global policy "Platinum treatment must be provided using server subtype A-1" would intrude into the Compute subsystem by redefining the type of server to be used for a particular service treatment. While one could argue that such global policy should not be permitted, this is an event that requires detection and proper resolution. A possible resolution is for the Compute subsystem to import the more restrictive policy into its local database. The original local policy would remain in the database as is along with the new restrictive policy. The local policy engine would then enforce the more restricted form of the policy after this policy change, which could make already existing resource allocations non-compliant and requiring corrective actions, e.g., Platinum treatment being currently provided by a server of type A instead of a server of type A-1.

If the new Global policy read "Platinum treatment must be provided using server of types A or B" instead, the Compute subsystem would

not need to do anything different, since the Compute subsystem has a more restrictive local policy in place, i.e., "Platinum treatment must be provided using server of type A."

The above examples demonstrate the need for subsystems to subscribe to policy updates at the Global policy level. A policy publication/subscription (pub/sub) bus would be required as shown in Figure 4.

```
+---------------------------------------------------------------------+
|          +-------------------------------------------------+        |
|          |               Global Policy Engine          |        |
|          +-------------------------------------------------+        |
|                                                                     |
|          +-------------------------------------------------+        |
|          |                 Global Policies             |        |
|          +-------------------------------------------------+        |
+---------------------------------------------------------------------+
                                  ^
                                  |
                                  |
Policy Pub/Sub Bus                V
   -----------------------------------------------------------------
        ^                 ^                 ^                 ^
        |                 |                 |                 |
        |                 |                 |                 |
        V                 V                 V                 V
+-------------+   +-------------+   +-------------+   +-------------+
|Compute      |   |Network      |   |Storage      |   |Whatever     |
|Subsystem    |   |Subsystem    |   |Subsystem    |   |Subsystem    |
|             |   |             |   |             |   |             |
|Local Policy |   |Local Policy |   |Local Policy |   |Local Policy |
|Engine       |   |Engine       |   |Engine       |   |Engine       |
|             |   |             |   |             |   |             |
|Local        |   |Local        |   |Local        |   |Local        |
|Policies:    |   |Policies     |   |Policies     |   |Policies     |
| P0, P1,     |   | P0, P1,     |   | P0, P1,     |   | P0, P1,     |
|             |   |             |   |             |   |             |
+-------------+   +-------------+   +-------------+   +-------------+
```
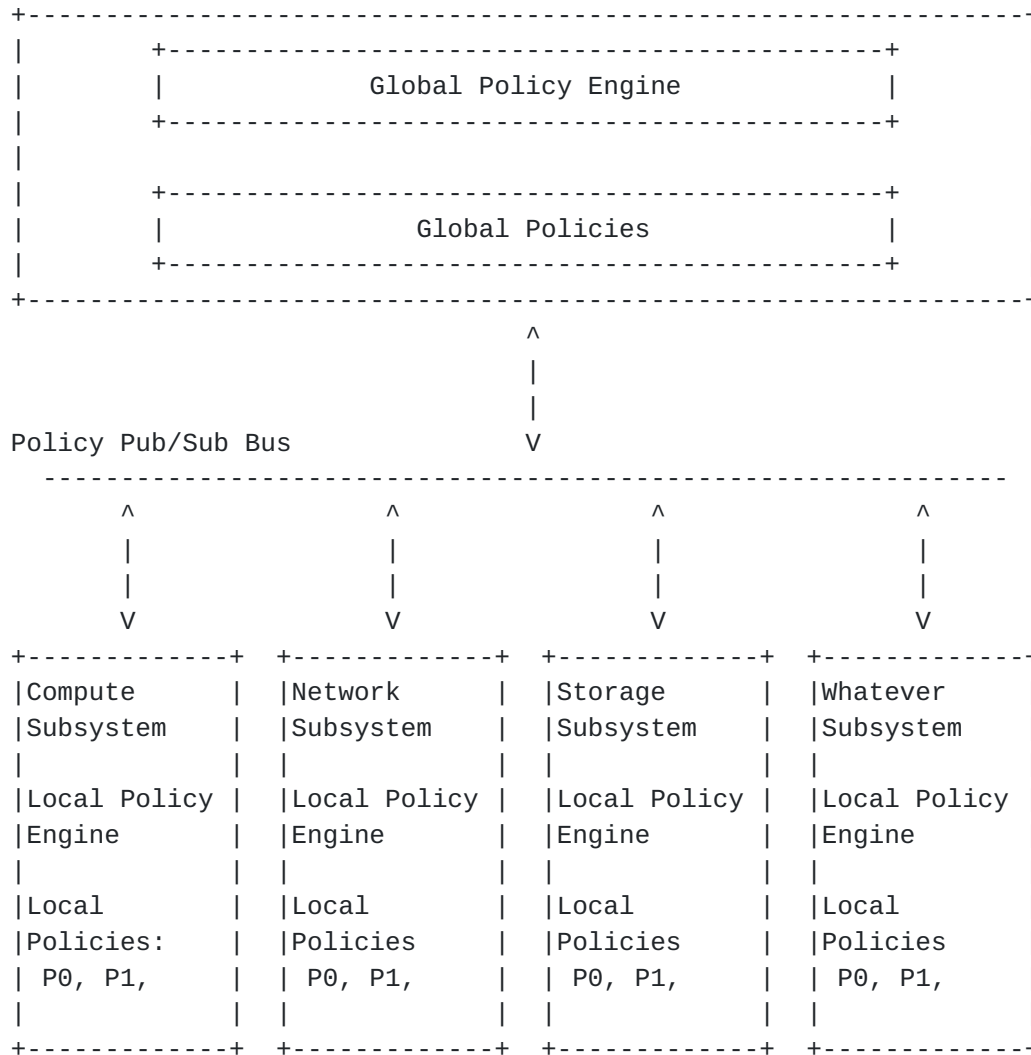
Figure 4: A Policy Pub/Sub Bus

A policy conflict may force policies to change scope. Consider the following existing policies in a Data Center:

Compute subsystem policy: "Platinum treatment requires a server of type A or B."

Storage subsystem policy: "Platinum treatment requires a server storage of type X or Y."

Now consider the outcome of adding the following new Global policy: "Platinum treatment requires a server of type A when storage of type X is used or a server of type B when storage of type Y is used."

This new Global policy intrudes into the Compute and Storage subsystems. Again, one could argue that such global policy should not be permitted. Nevertheless, this is an event that would require detection and proper resolution. This Global policy causes a conflict because the Compute and Storage subsystems can no longer independently define whether to use a server of type A or B or storage of type X or Y, respectively. If the Compute subsystem selects server of type A for a customer and the Storage subsystem selects storage of type Y for that same customer service the Global policy is violated. In conclusion, if such global policy is permitted, the Compute and Storage subsystems can no longer make such selections. A possible conflict resolution is for the Compute and Storage subsystems to relegate policy enforcement for such resources to the Global policy engine. In this example, the Global Policy engine would need to coordinate with the Compute and Storage subsystems the selection of appropriate resource types to satisfy that policy.

That suggests that the policy pub/sub bus should in fact be an integral part of the northbound service interfaces (NBI) of the subsystems in the hierarchy. Such issue was analyzed in [8], where the concepts of service capability, service availability, and service instantiation were introduced to enable a higher-level subsystem to properly select services and resources from lower-level subsystems to satisfy existing policies.

The above example demonstrates again the need for subsystems to subscribe to policy updates at the higher policy level (the Global policy level in this example) as shown in Figure 4.

If, as demonstrated, a Global policy may "hijack" or "nullify" local policies of subsystems, what exactly makes the scope of a policy local versus global then?

Proposition: A Local Policy does not affect the compliance state imposed by global Policies or the local policies of other subsystems.

The above non-exhaustive examples demonstrate that global and local policies may conflict in subtle ways. Policy conflicts will also policy framework requires a policy pub/sub bus between all levels to allow for conflict detection, conflict information propagation, and

conflict resolution (Figure 5).
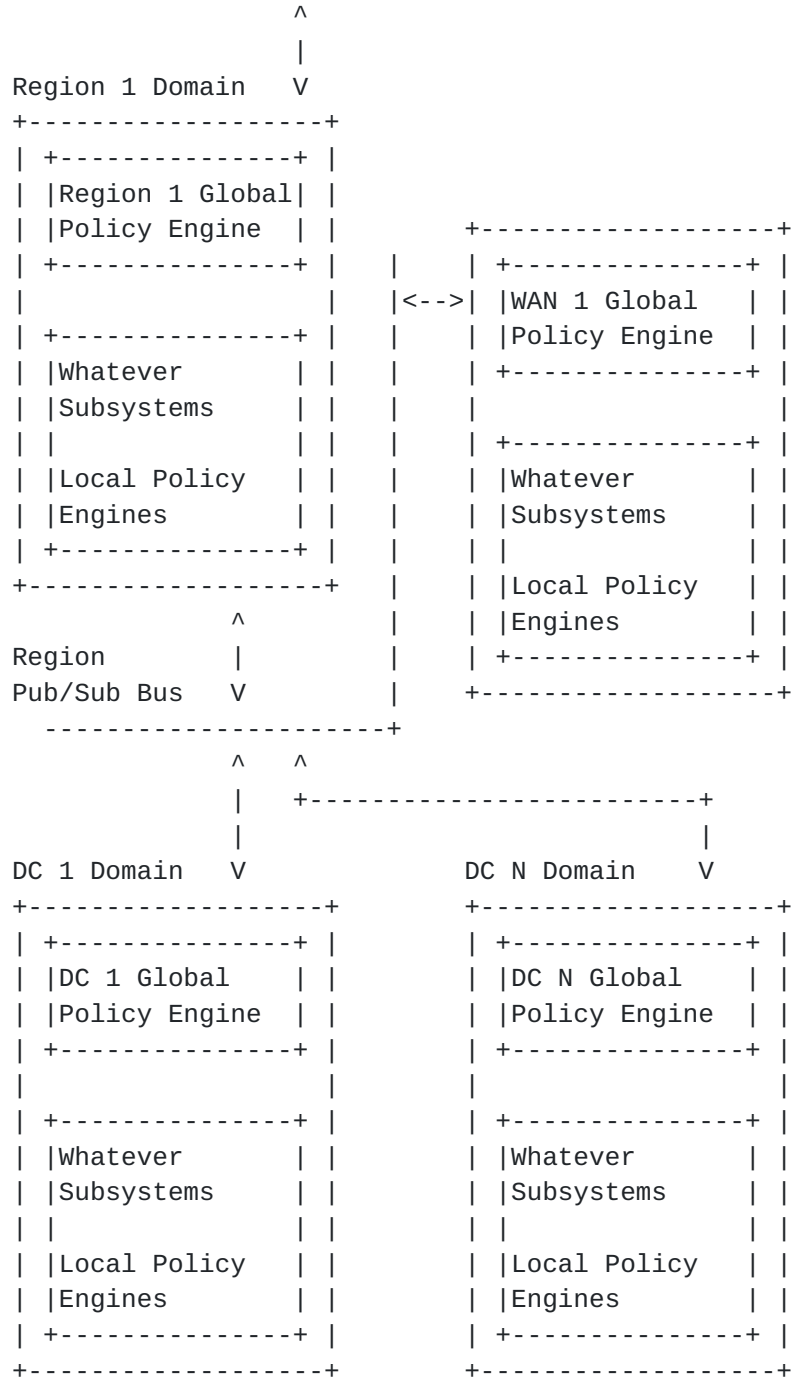
```
     Pub/Sub bus to higher level
                        ^
                        |
        Region 1 Domain    V
        +------------------+
        | +--------------+ |
        | |Region 1 Global| |
        | |Policy Engine  | |            +------------------+
        | +--------------+ |     |    | +--------------+ |
        |                  | |<-->|  | |WAN 1 Global   | |
        | +--------------+ |     |    | |Policy Engine  | |
        | |Whatever      | |     |    | +--------------+ |
        | |Subsystems    | |     |    |                  |
        | |              | |     |    | +--------------+ |
        | |Local Policy  | |     |    | |Whatever       | |
        | |Engines       | |     |    | |Subsystems     | |
        | +--------------+ |     |    | |               | |
        +------------------+     |    | |Local Policy   | |
                    ^            |    | |Engines        | |
        Region      |            |    | +--------------+ |
        Pub/Sub Bus    V         |    +------------------+
           ---------------------+
                    ^    ^
                    |   +------------------------+
                    |                            |
        DC 1 Domain    V            DC N Domain    V
        +------------------+        +------------------+
        | +--------------+ |        | +--------------+ |
        | |DC 1 Global   | |        | |DC N Global   | |
        | |Policy Engine | |        | |Policy Engine | |
        | +--------------+ |        | +--------------+ |
        |                  |        |                  |
        | +--------------+ |        | +--------------+ |
        | |Whatever      | |        | |Whatever      | |
        | |Subsystems    | |        | |Subsystems    | |
        | |              | |        | |              | |
        | |Local Policy  | |        | |Local Policy  | |
        | |Engines       | |        | |Engines       | |
        | +--------------+ |        | +--------------+ |
        +------------------+        +------------------+
```

              Figure 5: Pub/Sub Bus - Hierarchical Policy Framework

8. Examples

8.1 Establishment of a Multipoint Ethernet Service

Consider a service provider with an NFV infrastructure (NFVI) with
multiple vPoPs, where each vPoP is a separate administrative domain.
A customer "Z" requests the creation of a "multipoint Silver Ethernet
service" between three of its sites, which are connected to service
provider's vPoPs A, B, and C. The customer request is carried out
using a service provider self-service web portal, which offers
customers multiple service type options, e.g., point-to-point and
multipoint Ethernet services, and multiple service levels per service
type, e.g., Platinum, Gold, and Silver Ethernet services, where the
different service levels may represent different service
specifications in terms of QoS, latency, and etc. The web portal
relays the request to a service provider's OSS/BSS. The service
request is stored as a service policy that reads as: "multipoint
Silver Ethernet service between vPoPs A, B, and C for customer Z".

The OSS/BSS subsystem communicates the service request and
requirements as a policy to a global NFV Orchestrator (NFVO)
subsystem. The service provider's vPoP NFV infrastructure
architecture may vary depending on the size of each vPoP and other
specific needs of the service provider. For example, a vPoP may have
a local NFVO subsystem and one or more local Virtual Infrastructure
Manager (VIM) subsystems or it may simply have a VIM, but no local
NFVO subsystem. For simplicity of exposition, assume that the service
provider has a VIM and no NFVO per vPoP (as in Figure 6). In this
case, the global NFVO subsystem communicates the service request and
requirements as a policy to the VIMs of vPoPs A, B, and C.

At each vPoP, the local VIM will carry out the requested service
policy based on the local configuration of respective subsystems and
current availability of resources. For example, Silver service type,
as specified in the policy, may translate in vPoP A to use a specific
vCPE VNF type, say vCPE_X, while Silver service type in vPoP B may
translate to a different vCPE VNF type, say vCPE_Y, due to local
subsystem configurations (refer to Section 2 for a discussion on
subsystem actions and configurations). Similarly, the local VIM
interaction with the vPoP's compute, network, and storage subsystems
may lead to local configurations of these subsystems driven by the
translation of the policy by the respective subsystems (see Section 3
for a discussion on global versus local policies).

The global NFVO subsystem would potentially communicate the service
policy to a WAN infrastructure management (WIM) subsystem (not shown
in Figure 6), to provision a multipoint Silver Ethernet service
between vPoPs A, B, and C. The WIM subsystem could oversee a

hierarchy of other subsystems, e.g., SDN multi-domain architecture of controllers deployed as a hierarchy of network regions (see [8]). Network subsystems would translate the Silver type requirement to a local configuration (again, refer to Section 2 for a discussion on subsystem actions and configurations).

As depicted in Figure 6, service policy communications should employ a policy pub/sub bus between the subsystems' policy engines in the policy hierarchy (see Section 6 for a discussion on policy pub/sub bus). The global NFVO subsystem should have visibility into the policies defined locally at each vPoP to be able to detect any potential global policy conflicts, e.g., a local vPoP administrator could add a local policy that violates or conflicts with a global policy. In addition, the global NFVO subsystem would benefit from being able to import the currently configured services at each vPoP. For example, each vPoP could publish a table of currently configured services. The global NFVO would use such information to monitor global policy conformance and also to facilitate detection of policy violations when new global policies are created, e.g., a global level administrator is about to add a new global policy that, if committed, would make certain already configured services a violation of the policy. The publication of subsystem service tables for consumption by a global policy engine is a concept used in the Congress [19] OpenStack [18] project.

In the hierarchical policy framework described in this document, a subsystem's currently configured services table could be published to higher tier policy engines using the policy pub/sub bus. The subsystem's currently configured services table would describe configured services based on the configured policy name space for the respective policy pub/sub bus. The name space of a policy pub/sub bus refers to the name space available to communicate policies between the subsystems connected to the particular policy pub/sub bus. In this example, Ethernet services use the name space "Platinum", "Gold", and "Silver". A policy can then specify Silver Ethernet service. The policy name space would be an attribute associated with a particular policy pub/sub bus and should be pre-defined/pre-configured in the respective subsystems for each policy pub/sub bus.

Note that in a hierarchical policy framework a policy engine may use more than one policy pub/sub bus, e.g., a policy pub/sub bus to communicate with higher tier policy engines and another policy pub/sub bus to communicate with lower tier policy engines. For example, Figure 6 shows a policy pub/sub bus between the global NFVO subsystem and the vPoPs. Each vPoP could have an internal hierarchy of policy engines, e.g., VIM policy engine communicating with network (e.g., SDN controller), compute (VM orchestration), and storage subsystems' policy engines.

The amount of information that would be reported by a subsystem's currently configured services table would then depend on the pre-defined name space of a particular policy pub/sub bus. In this example, the vPoPs would communicate that customer Z is assigned/configured a vCPE VNF of type Silver.

Note that the described policy framework does not mandate or preclude use of detailed name spaces. However, to promote scalability and limit complexity, one should preferably use a name space hierarchy where the name spaces used by higher tier policy engines would be limited to higher level details. For example, suppose that vPoP A supports VNF types Silver.vCPE_X1 and Silver.vCPE_X2, that is, vCPE_X1 and vCPE_X2 are VNFs that were configured at vPoP A as supporting Silver services. Local policies in vPoP A would be used for the selection of vCPE_X1 or vCPE_X2 VNF when a service request requires a Silver vCPE VNF. vPoP A would report customer Z as using Silver.vCPE_X1 vCPE VNF (instead of simply Silver vCPE VNF) only when the name space between vPoP A and the global NFVO defines this granularity of Ethernet services. Note that one would want to define Silver.vCPE_X1 and Silver.vCPE_X2 as part of the policy name space between vPoP A and the global NFVO if the capability to specify such policy specificity is desired at the global level. However, the higher the degree of specificity allowed at the higher tiers of the policy hierarchy the higher the operational complexity.
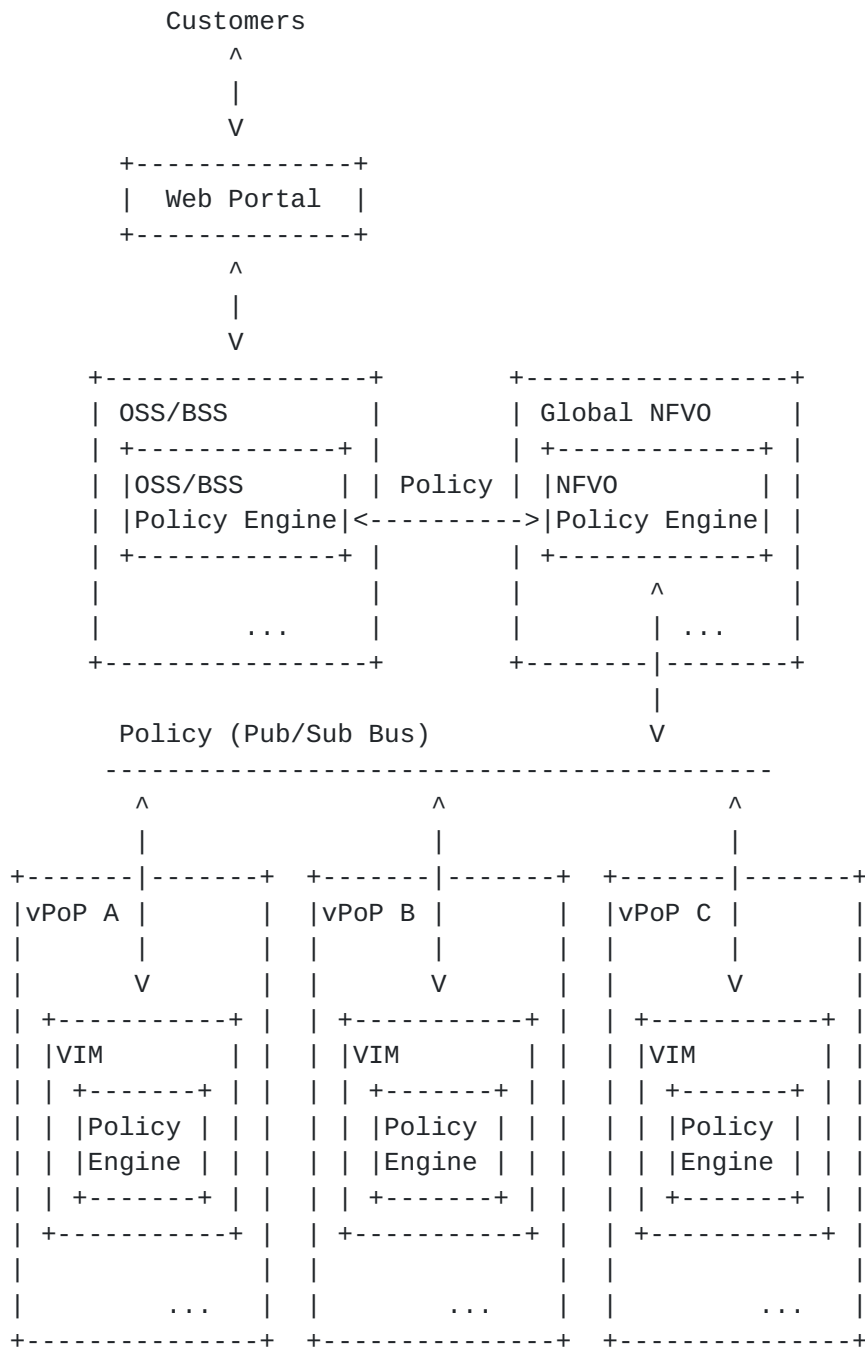
```
                    Customers
                       ^
                       |
                       V
            +--------------+
            |  Web Portal  |
            +--------------+
                    ^
                    |
                    V
       +-----------------+           +-----------------+
       | OSS/BSS         |           | Global NFVO     |
       | +-------------+ |           | +-------------+ |
       | |OSS/BSS      | | | Policy | |NFVO         | |
       | |Policy Engine|<----------->|Policy Engine| |
       | +-------------+ |           | +-------------+ |
       |                 |           |       ^         |
       |           ...   |           |       | ...     |
       +-----------------+           +--------|--------+
                                              |
         Policy (Pub/Sub Bus)                 V
         ----------------------------------------------
            ^                    ^                  ^
            |                    |                  |
    +-------|-------+    +-------|-------+   +-------|-------+
    |vPoP A |       |    |vPoP B |       |   |vPoP C |       |
    |       |       |    |       |       |   |       |       |
    |       V       |    |       V       |   |       V       |
    | +-----------+ |    | +-----------+ |   | +-----------+ |
    | |VIM        | |    | |VIM        | |   | |VIM        | |
    | | +-------+ | |    | | +-------+ | |   | | +-------+ | |
    | | |Policy | | |    | | |Policy | | |   | | |Policy | | |
    | | |Engine | | |    | | |Engine | | |   | | |Engine | | |
    | | +-------+ | |    | | +-------+ | |   | | +-------+ | |
    | +-----------+ |    | +-----------+ |   | +-----------+ |
    |               |    |               |   |               |
    |           ... |    |           ... |   |           ... |
    +---------------+    +---------------+   +---------------+
```

       Figure 6: Simplified view of a service provider's NFV Architecture:
                 Multipoint Ethernet Service Example

8.2 Policy-Based NFV Placement

IRTF draft [10] describes a detailed example of a global policy written in Datalog [1] applicable to compute to promote energy conservation for the NFVIaaS use case [4] in an OpenStack framework. The goal of that policy is to address the energy efficiency requirements described in the ETSI NFV Virtualization Requirements [5].

Related to the above, energy efficiency using analytics-driven policies in the context of OpenStack Congress [19] policy as a service was presented and demonstrated at the Vancouver OpenStack summit [11], where the Congress policy engine delegates VM placement to a VM placement engine that migrates under-utilized VMs to save energy.

9. Summary

This document approached the policy framework and architecture from the perspective of overall orchestration requirements for services involving multiple subsystems. The analysis extended beyond common orchestration for compute, network, and storage subsystems to also include energy conservation constraints. This document also analyzed policy scope, global versus local policies, policy actions and translations, policy conflict detection and resolution, interactions among policies engines, and a hierarchical policy architecture/framework to address the demanding and growing requirements of NFV environments, applicable as well to general cloud infrastructures.

The concept of NFV and the proposed policy architecture is applicable to service providers and also enterprises. For example, an enterprise branch office could have capacity and energy constraints similar to that of many service provider NFV vPoPs in constrained environments. This is an aspect that would be worth examining in detail in future work.

10. IANA Considerations

This draft does not have any IANA considerations.

11. Security Considerations

Security issues due to exchanging policies across different administrative domains are an aspect for further study.

[12](#). References

[12.1](#). Normative References

[12.2](#). Informative References

[1] Ceri, S. et al., "What you always wanted to know about Datalog (and never dared to ask)," IEEE Transactions on Knowledge and Data Engineering, (Volume: 1, Issue: 1), August 2002

[2] "Common Information Model (CIM)," DTMF, http://www.dmtf.org/standards/cim

[3] ETSI NFV White Paper: "Network Functions Virtualisation, An Introduction, Benefits, Enablers, Challenges, & Call for Action," http://portal.etsi.org/NFV/NFV_White_Paper.pdf

[4] ETSI GS NFV 001 v1.1.1 (2013-10): "Network Function Virtualisation (NFV); Use Cases," http://www.etsi.org/deliver/etsi_gs/NFV/001_099/001/01.01.01_60/gs_NFV001v010101p.pdf

[5] ETSI GS NFV 004 v1.1.1 (2013-10): "Network Function Virtualisation (NFV); Virtualization Requirements," http://www.etsi.org/deliver/etsi_gs/NFV/001_099/004/01.01.01_60/gs_NFV004v010101p.pdf

[6] ETSI GS NFV-INF 001 v.1.1.1 (2015-01): "Network Function Virtualisation (NFV); Infrastructure Overview," http://www.etsi.org/deliver/etsi_gs/NFV-INF/001_099/001/01.01.01_60/gs_NFV-INF001v010101p.pdf

[7] ETSI GS NFV 002 v1.2.1 (2014-12): "Network Function Virtualisation (NFV); Architectural Framework," http://www.etsi.org/deliver/etsi_gs/NFV/001_099/002/01.02.01_60/gs_nfv002v010201p.pdf

[8] Figueira, N. and Krishnan, R., "SDN Multi-Domain Orchestration and Control: Challenges and Innovative Future Directions," CNC VIII: Cloud and Multimedia Applications, IEEE International Conference on Computing (ICNC), February 2015

[9] Grit, L. et al., "Virtual Machine Hosting for Networked Clusters: Building the Foundations for "Autonomic" Orchestration," Virtualization Technology in Distributed Computing, 2006. VTDC 2006.

[10] Krishnan, R. et al., "NFVIaaS Architectural Framework for Policy Based Resource Placement and Scheduling," https://datatracker.ietf.org/doc/draft-krishnan-nfvrg-policy-based-rm-nfviaas/

[11] Krishnan, R. et al., "Helping Telcos go Green and save OpEx via Policy", Talk and demo at the Vancouver OpenStack summit. Video Link: https://www.openstack.org/summit/vancouver-2015/summit-videos/presentation/helping-telcos-go-green-and-save-opex-via-policy

[12] Lee, S. et al., "Resource Management in Service Chaining", https://datatracker.ietf.org/doc/draft-irtf-nfvrg-resource-management-service-chain/

[13] Moore, B., et al., "Information Model for Describing Network Device QoS Datapath Mechanisms," RFC 3670, January 2004

[14] Moore, B. et al., "Policy Core Information Model -- Version 1 Specification," RFC 3060, February 2001

[15] "OpenDaylight Group Based Policy," https://wiki.opendaylight.org/view/Project_Proposals:Group_Based_Policy_Plugin

[16] "OpenDaylight Network Intent Composition Project," https://wiki.opendaylight.org/index.php?title=Network_Intent_Composition:Main#Friday_8AM_Pacific_Time

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

[18] "OpenStack, "http://www.openstack.org/

[19] "OpenStack Congress, "https://wiki.openstack.org/wiki/Congress

[20] "OpenStack Neat, "http://openstack-neat.org/

[21] "OpenStack Neutron, "https://wiki.openstack.org/wiki/Neutron

[22] "Policy Framework Working Group," IETF, http://www.ietf.org/wg/concluded/policy.html

[23] Westerinen, A. et al., "Terminology for Policy-Based Management," RFC 3198, November 2001

Authors' Addresses

   Norival Figueira
   Brocade Communications Systems, Inc.
   nfigueir@Brocade.com

   Ram (Ramki) Krishnan
   Dell
   Ramki_Krishnan@Dell.com

   Diego R. Lopez
   Telefonica I+D
   diego.r.lopez@telefonica.com

   Steven Wright
   AT&T
   sw3588@att.com

   Dilip Krishnaswamy
   IBM Research
   dilikris@in.ibm.com