Internet Research Task Force (IRTF) N. Figueira
Internet Draft R. Krishnan
Category: Informational Brocade

Expires: May 2015 November 25, 2014

Policy Architecture and Framework for NFV and Cloud Services

draft-norival-nfvrg-nfv-policy-arch-00

Abstract

A policy architecture and framework is discussed to support NFV and Cloud 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 analysis extends beyond compute, network, and storage subsystems to also include energy conservation. This document also analyses 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 and Cloud environments.

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) [1] and Cloud services. In these environments, policies are used to enforce business rules and to specify resource constraints in a number of

subsystems, e.g., compute, storage, network, energy, and etc., and across subsystems.

The current work in the area of policy for NFV and Cloud services is typically focused on individual subsystems and addresses very specific use cases or environments. For example, [1] addresses network subsystem policy for network virtualization, [3] proposes an open source project in the area of network policy as part of the OpenDaylight software define networking (SDN) controller framework [4], [5] specifies an information model for network policy, [6] focuses on placement and migration policies for distributed virtual computing, [7] is an open source project proposal in OpenStack [10] to address policy for cloud environments.

This document approaches policy, policy framework, and policy architecture for NFV and Cloud services from the perspective of overall orchestration requirements for services involving multiple subsystems. The analysis extends beyond compute, network, and storage subsystems to also include energy conservation as it applies to NFV, Cloud, and other environments. The analysis in this document extends beyond a single data center (DC) or administrative domain to include multiple data centers and networks forming hierarchical domain architectures [22].

This focus of document is not general policy theory, which has been intensively studied and documented on numerous publications over the past 10 to 15 years (see [5], [14], [16], [17], and [18] to name a few). This document's purpose is to discuss and document a policy architecture (using known policy concepts and theories) to address the unique requirements of NFV and Cloud services including multiple data centers and networks forming hierarchical domain architectures [22].

With the above goals, this document analyses policy scope, global versus local policies, policy actions and translations of actions, policy conflict detection and resolution, the interactions among policies engines of the different data center and network subsystems, and a hierarchical policy architecture/framework to address the demanding and growing requirements of NFV and Cloud environments.

2. Policy Statement and Actions

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 a VNF X]" 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 subsystem 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) 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., [14], [17], [15], and [16]. In this document, such device configurations are called policy enablement/enforcement technologies to set them apart from the actually intended policy, i.e., "a given customer must be given platinum treatment" in the above example.

The translation of a policy into appropriate subsystem configurations requires additional information that is usually subsystem dependent and technology dependent. Therefore, policies should not be written in terms of policy enablement/enforcement technologies. Policies should be translated at the subsystems using the appropriate policy enablement/enforcement technologies employed by the subsystems. Figure 1 provides a few examples where the policy "a given customer must be given platinum treatment" is translated to appropriate configurations at the respective subsystems.

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

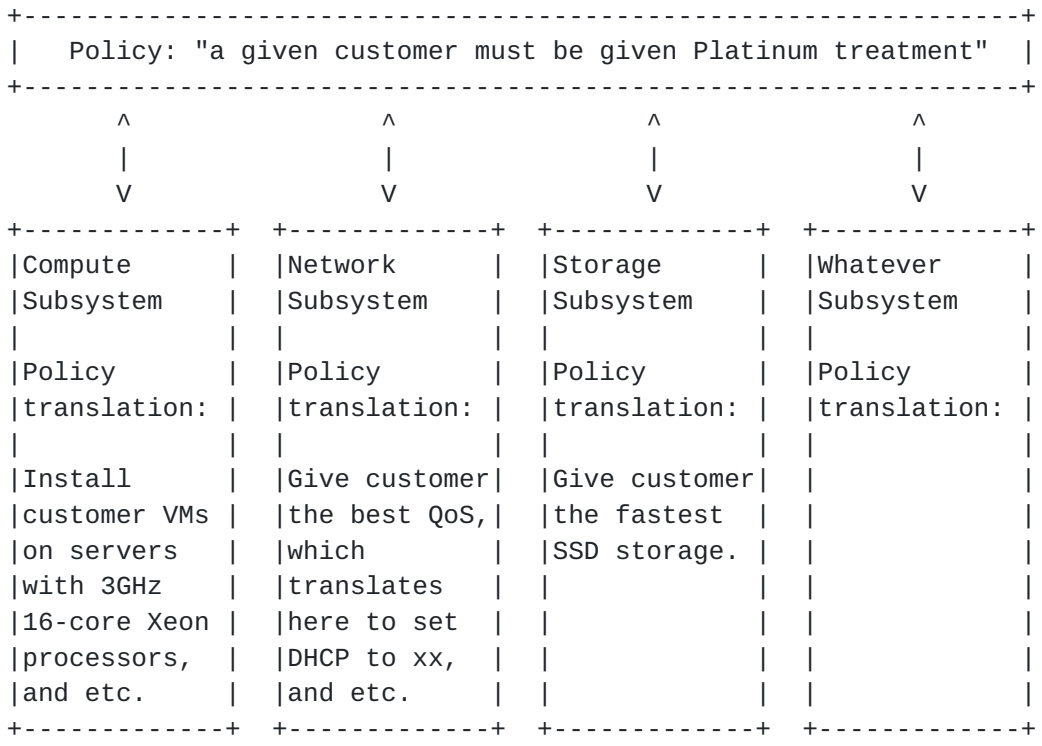

Figure 1: Example of Subsystem Translations of Policy Actions

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 sub-systems. As shown in Figure 1, platinum treatment could be translated to servers of a given performance specification in a compute subsystem and storage of a given performance specification (SSD in this example) 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 by a local policy engine of the respective subsystems.

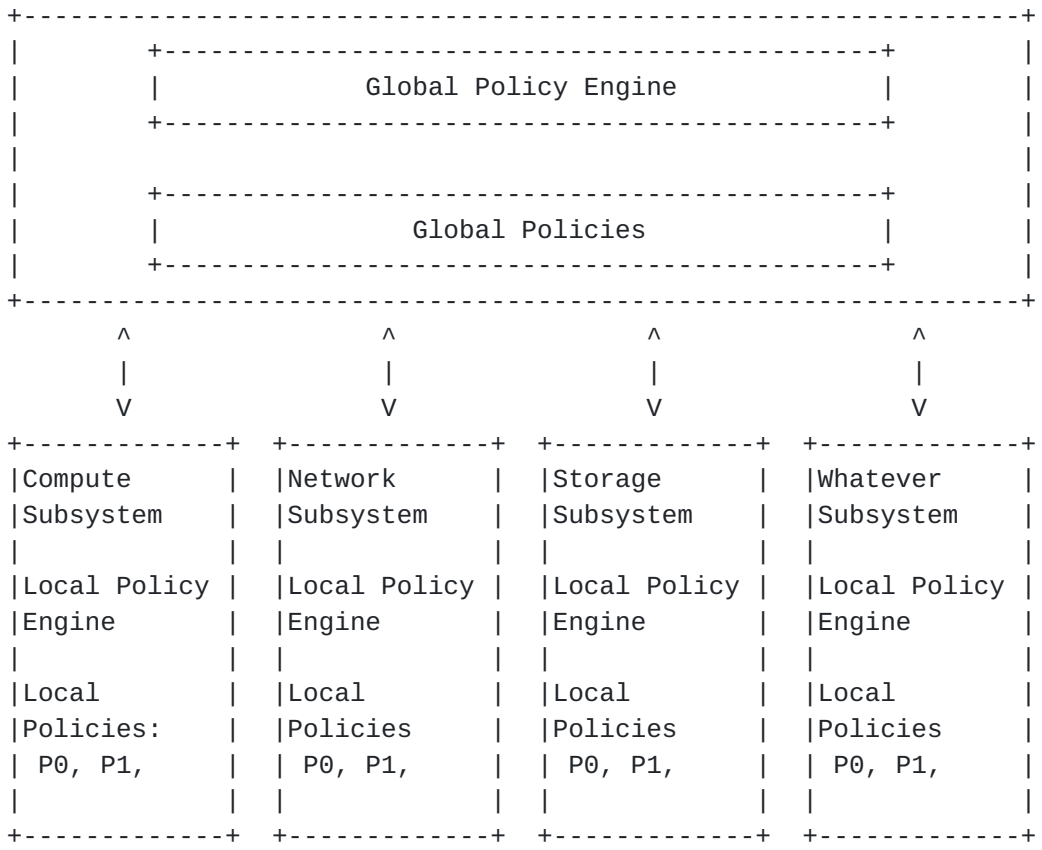

Figure 2: Global versus Local Policy Engines

4. 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:

- o SDN Controllers, e.g., OpenDaylight [4].
- o OpenStack [10] components such as, Neutron, Cinder, Nova, and etc.
- o Directories, e.g., LDAP, ActiveDirectory, and etc.
- o Applications in general, e.g., Apps on top of OpenDaylight or OpenStack and standalone Apps.
- o Physical and virtual network elements, e.g., routers, firewalls, ADCs, and etc.
- o Energy, e.g., OpenStack Neat [8]

Therefore, a policy framework may involve a multitude of subsystems. Subsystems may include other lower level subsystems, e.g., Neutron [9] 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.

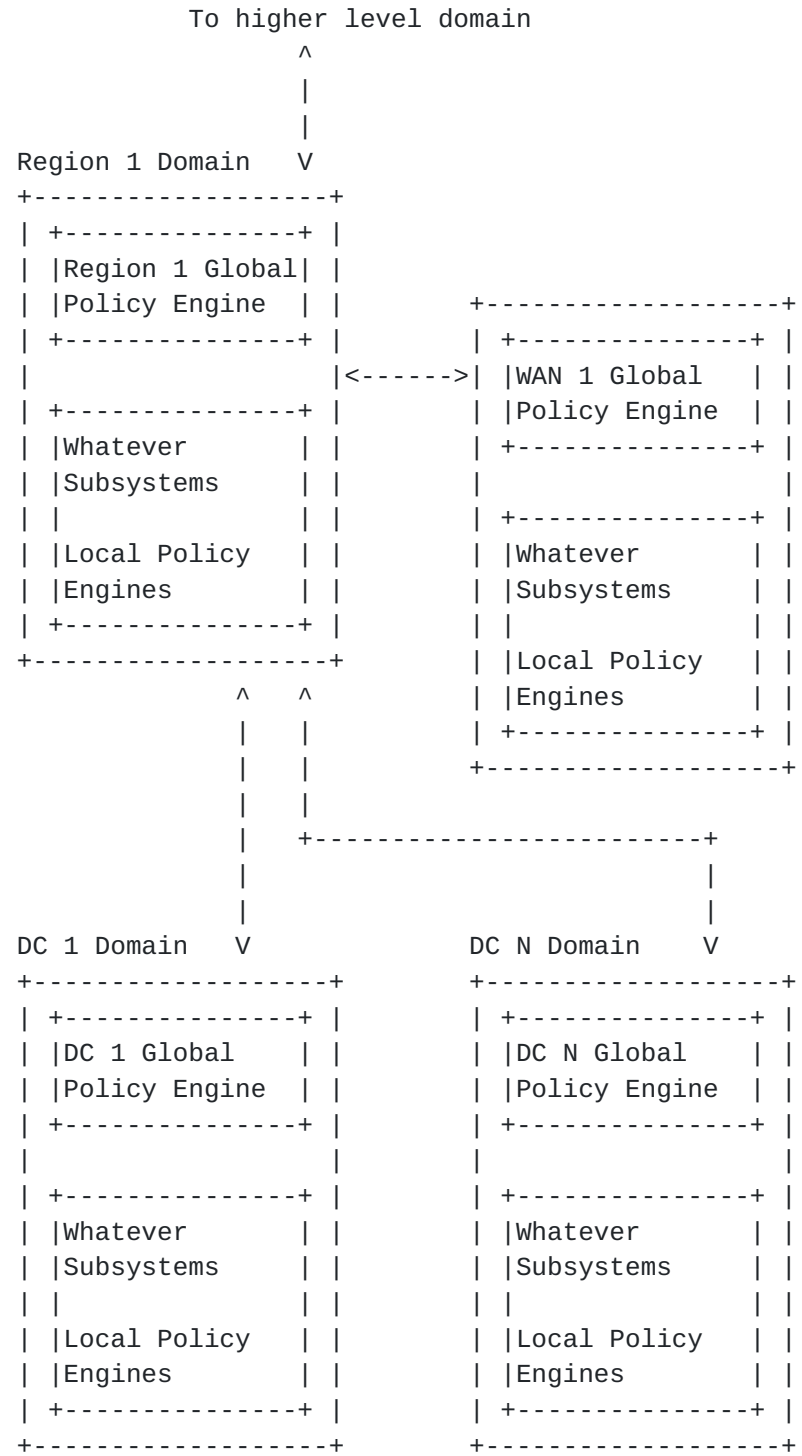

Figure 3: A Hierarchical Policy Framework

5. 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 with batched policies, 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 a service provider defines separate administrative domains. The region administration could define batched polices to be pushed to the Compute subsystem of a Data Center. 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 in the region administrative domain. Thus, there is a need for a reactive policy conflict resolution mechanism besides preemptive techniques.

6. 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 as is along with the new restrictive policy. The local policy engine would enforce the more restricted form of the policy. This could make already existing resource allocations non-compliant and requiring corrective actions. If the new Global policy reads "Platinum treatment must be provided using server of types A or B", the Compute subsystem would not need to do anything, since the Compute subsystem has a more restrictive local policy in place.

The above example demonstrates the need for subsystems to subscribe to policy updates at the Global policy level. Some sort of policy publication/subscription (pub/sub) bus would be required as shown in Figure 4.

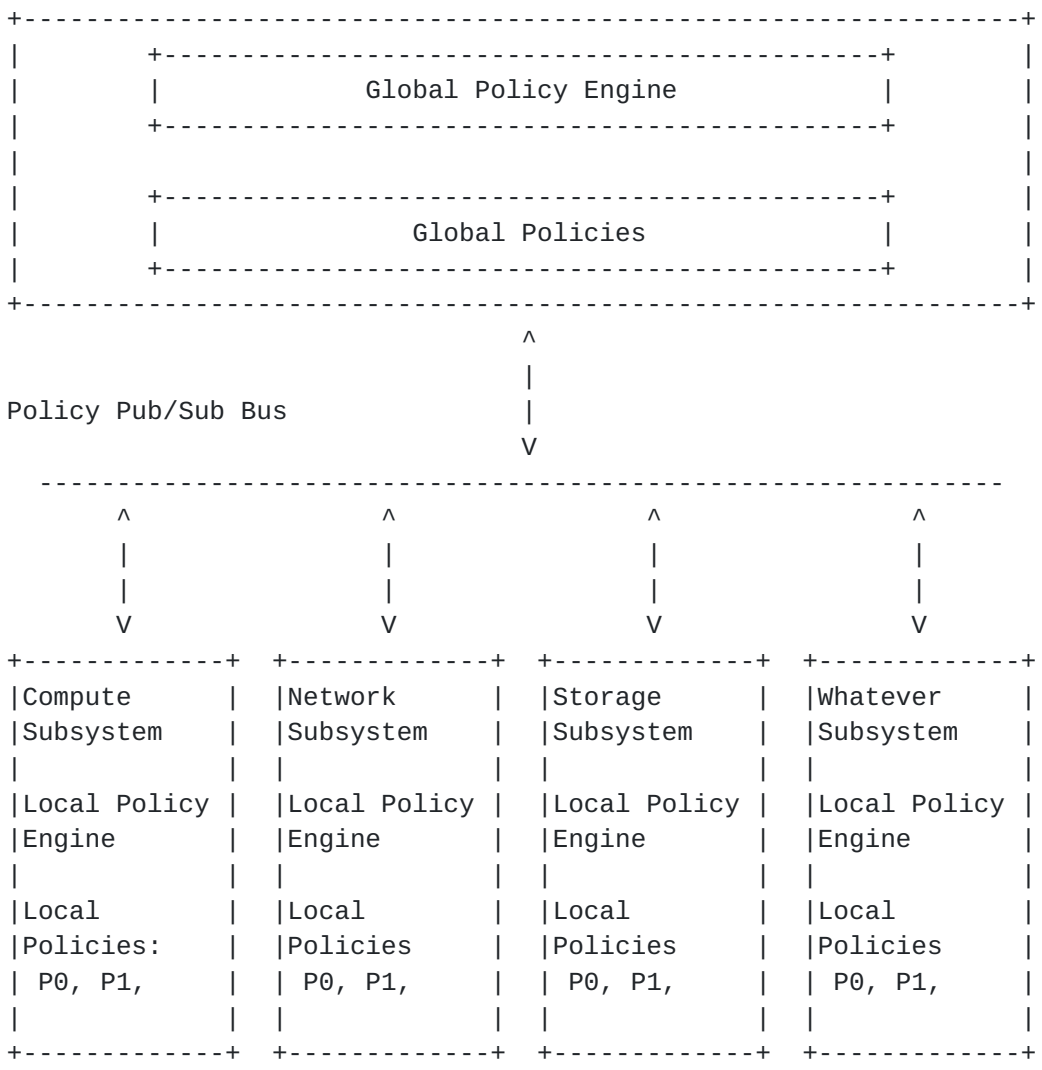

Figure 4: A Policy Pub/Sub Bus

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 requires 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. 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.

The above example demonstrates again the need for subsystems to subscribe to policy updates at the Global policy level 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 arise in deeper hierarchical policy frameworks. A hierarchical policy framework requires a policy pub/sub bus between all levels to allow for conflict detection and resolution (Figure 5).

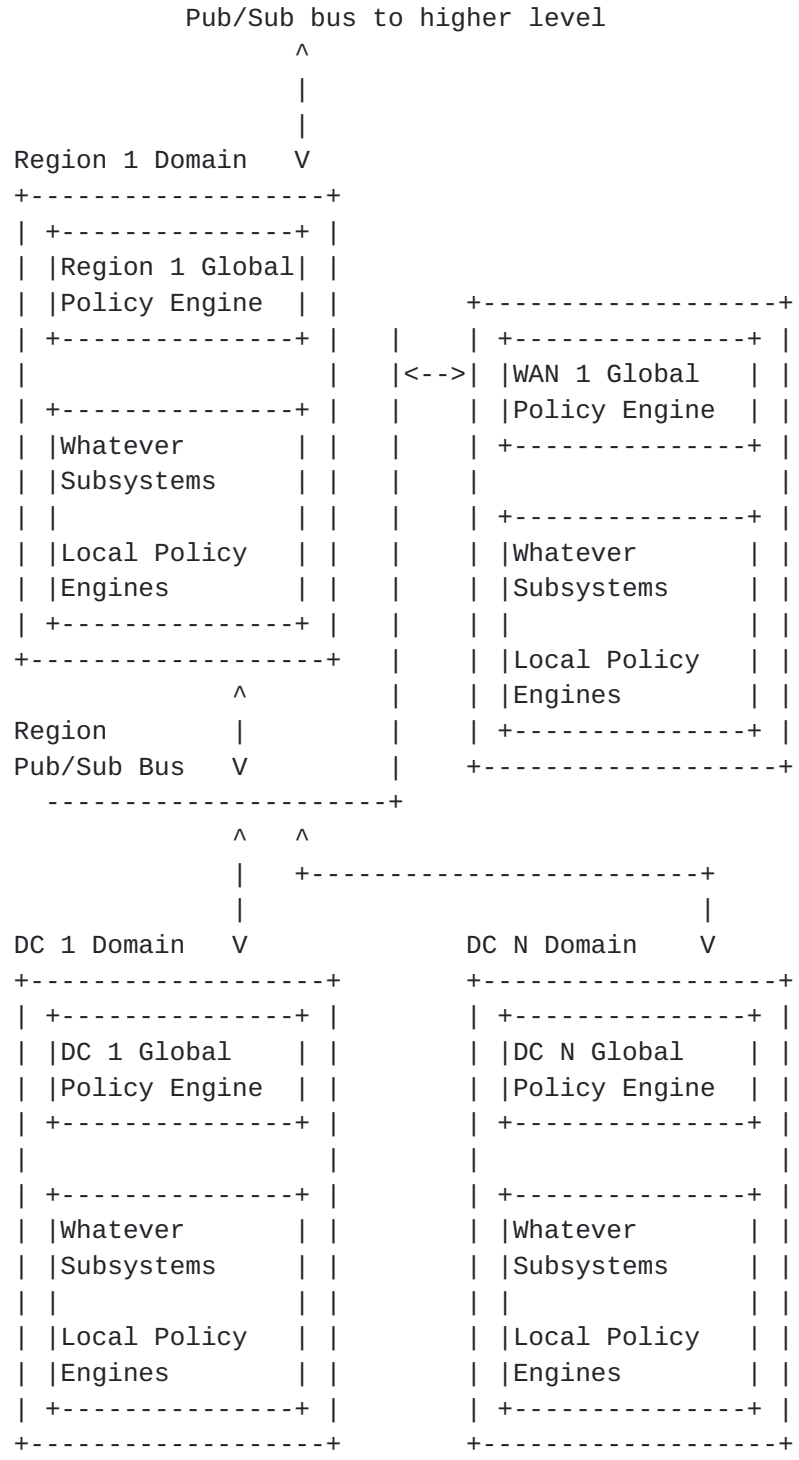

Figure 5: Pub/Sub Bus - Hierarchical Policy Framework

7. Example of Policy-based NFV Placement

IRTF draft [19] describes a detailed example of a global policy in Datalog [18] applicable to compute and energy sub-systems for the NFVIaaS use case [20] in an OpenStack framework. The goal of this policy is to address the energy efficiency requirements described in the ETSI NFV Virtualization Requirements [21].

8. 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 compute, network, and storage subsystems to also include energy conservation. 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 and Cloud environments.

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 a service provider mini NFV DC. This is an aspect which would be worth examining in detail in future work.

9. IANA Considerations

This draft does not have any IANA considerations.

10. Security Considerations

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

11. Acknowledgements

The authors would like to acknowledge Steven Wright, Uwe Michel, Klaus Martiny, Diego Lopez, Pedro Andres Aranda Gutierrez, Dilip Krishnaswamy and Tim Hinrichs for the discussions on some of these topics.

12. References

12.1. Normative References

12.2. Informative References

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

[2] Rahman, M. R. et al., "SVNE: Survivable Virtual Network Embedding Algorithms for Network Virtualization, "IEEE Transactions on Network and Service Management,(Volume: 10 , Issue: 2)

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

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

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

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

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

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

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

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

[11] "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

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

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

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

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

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

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

[18] 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

[19] 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/

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

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

[22] "SDN Multi-Domain Orchestration and Control: Challenges and Innovative Future Directions," IEEE International Conference on Computing, Networking and Communications (ICNC) 2015 Workshop - Accepted

Authors' Addresses

Norival Figueira
Brocade Communications
nfigueir@Brocade.com

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