SUPA
Internet-Draft
Intended status: Informational
Expires: September 14, 2017

Y. Cheng
China Unicom
D. Liu
Alibaba Group
B. Fu
China Telecom
D. Zhang
Freelancer
N. Vadrevu
VN Telecom Consultancy

March 13, 2017

Applicability of SUPA
draft-cheng-supa-applicability-01

Abstract

SUPA will define a generic policy model, an imperative ECA (Event Condition Action) policy information model and a declarative (intent-based) policy information model which is the extension of the generic model, and a set of policy data models which will make use of the common concepts defined in the generic model. This memo will explore some typical use cases and demonstrate the applicability of SUPA policy models.

Status of This Memo

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

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

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

This Internet-Draft will expire on September 14, 2017.

Table of Contents

1. Introduction

One of the ways for network service automation is using network management and operation software applications. The applications may not be able to directly communicate with each network element; a hierarchical and extensible framework should be considered to hide the protocol specific and/or vendor specific details, high level network and service abstraction, and standardized programming API will be necessary.

SUPA will define policy generic models and data models, for service management and operation applications. [I-D.ietf-supa-generic-policy-info-model] defines a common set of concepts for various data models which may use different languages, protocols, and repositories. The generic policy information model (GPIM)[I-D.ietf-

supa-generic-policy-info-model] is defined for use in network operations and management applications. The ECA Policy Rule Information Model (EPRIM) [I-D.ietf-supa-generic-policy- info-model] extends the GPIM to define how to build policy rules according to the event-condition-action paradigm. The GPIM and the EPRIM will both be translated into corresponding YANG modules that define policy concepts, terminology, and rules in a generic and interoperable manner; additional YANG modules may also be defined from the GPIM and/or EPRIM to manage specific Functions, see [I-D.ietf-supa-generic-policy-data-model].

The generic data models will be used for domain or service specific data model. And there is no interoperability requirement for domain specific data models. The interoperability is guaranteed at the generic data model level via the common concepts.

2. Terminology

DC Data Center

PCE Path Computation Element

SP Service Provider

SUPA Simplified Use of Policy Abstractions

VM Virtual Machine

VPC Virtual Private Cloud

3. Framework

The SUPA Policy-based Management Framework is described in [I-D.ietf-supa -policy-based-management-framework]. Figure 1 is copied from [I-D.ietf-supa-policy-based-management-framework], for clarity reasons.

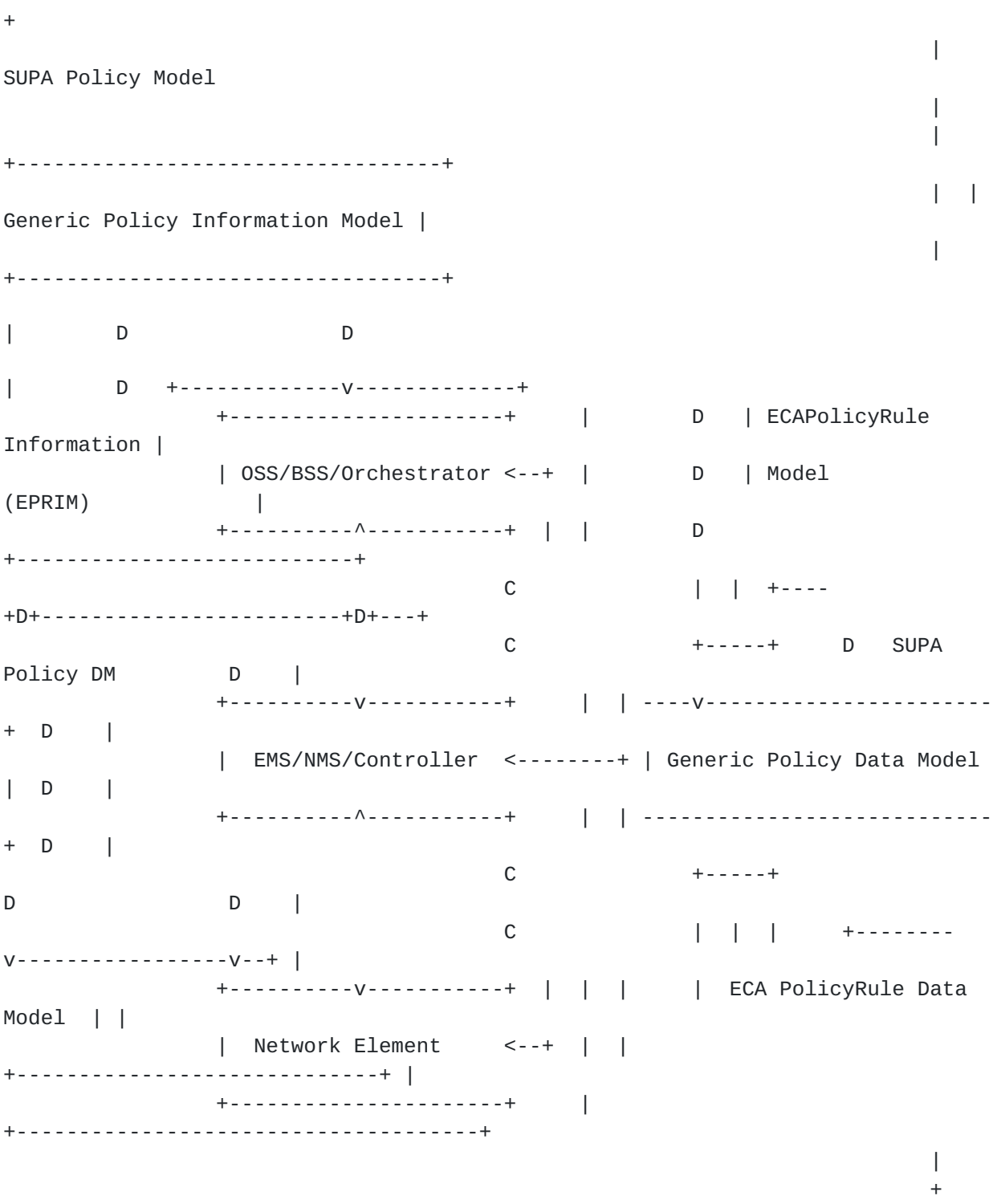

Figure 1: SUPA Policy Model Framework, copied from [I-D.ietf-supa-policy-based-management-framework]

In Figure 1:

The double-headed arrow with Cs means communication;

The arrow with Ds means derived from.

The components within this framework are:

SUPA Policy Model: represents one or more policy modules that contain the following entities:

Generic Policy Information Model: a model for defining policy rules that are independent of data repository, data definition, query, implementation languages, and protocol. This model is abstract and is used for design; it MUST be turned into a data model for implementation.

Generic Policy Data Model: a model of policy rules that are dependent on data repository, data definition, query, implementation languages, and protocol.

ECA Policy Rule Information Data Model (EPRIM): represents a policy rule as a statement that consists of an event clause, a condition clause, and an action clause. This type of Policy Rule explicitly defines the current and desired states of the system being managed. This model is abstract and is used for design; it MUST be turned into a data model for implementation.

ECA Policy Rule Data Model: a model of policy rules, derived from EPRIM, that consist of an event clause, a condition clause, and an action clause.

EMS/NMS/Controller: represents one or more entities that are able to control the operation and management of a network infrastructure (e.g., a network topology that consists of Network Elements).

Network Service and Resource Data Models: models of the service as well as physical and virtual network topology including the resource attributes (e.g., data rate or latency of links) and operational parameters needed to support service deployment over the network topology.

Network Element (NE), which can interact with local or remote EMS/NMS/Controller in order to exchange information, such as configuration information, policy enforcement capabilities, and network status.

As shown in Figure 1, SUPA will define generic policy models, which are independent of services and use cases. Policy data models can be derived from the generic models. The data model will define high level, maybe network-wide policies. Policy data model will be used in conjunction with service data models to generate configurations for network elements. The service data model is use case specific and will be developed by operators or third parties, which is out the scope of SUPA.

The service management applications will send SUPA data models to the service management system, where policy making and automated policy enforcement will be performed, and the data models will be mapped to configuration of network elements. Configuration of network elements is vendor specific, using various protocols, such Netconf, Restconf, etc.

SUPA also make use of information collected from network elements. The information may include warning or fault event, load status, traffic statistics, etc, which can be used to adjust network configurations. This kind of automation is done through ECA data models.

3.1. Network Manager/Controller

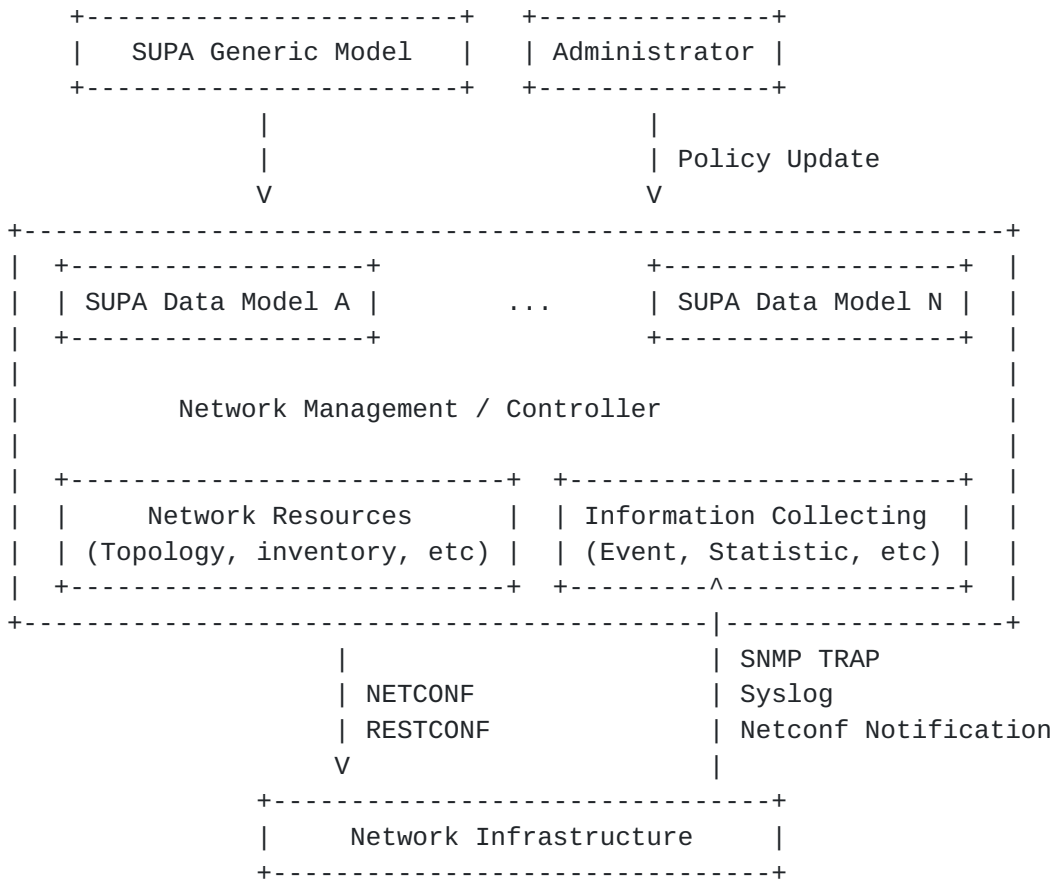

Figure 2: Network Manager / Controller

The internal details of the network manager / controller may be out of the scope of SUPA, but explaining how it works may help people to understand and implement SUPA.

Network administrator can send service deployment and management request to network manager / controller via SUPA data models. The data models will be converted into network elements configuration snippets. The configuration change may be performed instantly, or later triggered by events. The network manager / controller has the intelligence to decide which network devices should be configured, and what the configuration will be, which is derived from the actions specific in the data models explicitly or implicitly.

Network management related resources and information are stored in the network manager/controller, which contains the network topology (physical and virtual interconnection of network elements, etc), inventory (database of network elements, ports, device type, capabilities, etc.), protocol specific information, etc.

SUPA will make use of the existing work of other IETF WGs and other SDOs, such as if the topology data model is already defined in another IETF WG, SUAP will reference it rather than trying to define it again.

The network manager / controller will find out the list of network devices which should be configured for a specific demand or service.

For example, there is a configuration request:

All edge routers shall have SSH disabled.

An edge router is a router with connection to network(s) outside of the current network domain. The controller will query the topology database and find out all the routers with the attribute of "device-role == edge", or the controller may use more complicated algorithms to find out if a router is an edge route, which is implementation specific.

Similarly, another example is, the controller can make use of PCE engine to plan the links between DCs, and make sure the links are disjoint for better availability in case of failure. The PCE engine will be used in conjunction with the topology database to find out possible disjoint links.

The network manager / controller will also have other information, such as protocol specific information, traffic with TCP destination port 22 is SNMP traffic.

The network manager / controller also collect information from the network device, such events, logs, statistics, etc. The information may come from SNMP TRAP, Syslog, NETCONF notification, and other sources such as vendor specific protocols or extensions. The collected information may be used in conjunction with SUPA ECA data models for dynamic configuration change. An example use of the information is, if the load on a link between two DC exceeds a threshold, and there are multiple disjoint links between the two DCs, traffic steering will be triggered.

Event: link_load > threshold

Condition: there are disjoint links

Action: perform traffic steering

Some of the events are already standardized, such SNMP TRAP and NETCONF notification; some are implementation specific.

SUPA data models explicitly or implicitly specify network actions, and the actions may be expanded into more detail actions if necessary, and finally converted into protocol specific, vendor specific network element configuration snippets.

In the previous example shown below again:

All edge routers shall have SSH disabled.

The action in this case is "disable SSH traffic", the network manager / controller should converted this action into configuration "disable traffic on TCP port 22" in the IP stack, or an ACL rule which will drop traffic with TCP destination port 22.

The network manager / controller can support various types of southbound interface, such as NETCONF, RESTCONF, SNMP, OpenFLow, etc, which make it possible to support devices from different vendors. This is implementation specific and out of the scope of SUPA.

4. Use Cases of SUPA

4.1. Use Case 1: SNMP blocking

4.1.1. Introduction

This example will illustrate how to use the SUPA information model to block inbound and outbound SNMP traffic.

The following exemplar policy was posted to the SUPA mailing list:

ensures that SNMP is blocked on ports at the edge

of the administrative domain to prevent SNMP going

out or coming in from outside the enterprise. (1)

While this is simple for a human to understand, it is actually quite difficult for a machine to understand in its original form. This is because:

1) the text must be translated to a form that the device can understand

2) the nature of the policy is not clear (due to the inherent ambiguity of English)

4.1.2. Solution Approach

First, let's assume the following context:

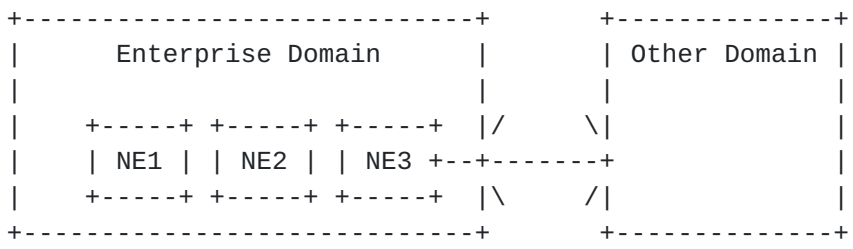

Figure 3: Blocking inbound and outbound SNMP traffic

In the above example, the only "edge" interface is that of NE3. This enables us to simplify (1) to:

block SNMP on NE3 (2)

This assumes that NE3 exists and is operational. This is a **big** assumption. This leads to the observation that in both (1) and (2), there are at least two different interpretations for each:

1) apply a set of actions directly to a SUPAPolicyTarget, assuming that the SUPAPolicyTarget understands SUPAPolicies, or

2) apply a set of desired actions that are already translated to a form that a SUPAPolicyTarget can understand

Note that a SUPAPolicyTarget could be the network device or a proxy for the network device.

The difference between these interpretations is whether a SUPAPolicy applies one or more SUPAPolicyActions **directly** to a SUPAPolicyTarget (that is without translation to, for example, CLI or YANG) versus whether a SUPAPolicy, as part of its action(s), produces something that the device (or its proxy) can understand.

Put another way, the first alternative shows how SUPAPolicies can directly control behavior, while the second alternative shows how a SUPAPolicy can invoke a set of actions that the device (or its proxy) can understand. Thus, policy (1) can be formulated as either:

- IF any network element has a port that meets the criterion of the role "edge interface", AND it is inside the EnterpriseDomain, then block SNMP traffic (3)

- IF a network element is added within the EnterpriseDomain

IF any of its ports take on the role "edge interface"

Add a filter to block SNMP traffic for that port (4)

The first case is the simplest, and likely what most people thought. Conceptually, it could look as follows:

Event: SNMP traffic is sent or received

Condition: IF this port implements the "edgeInterface" role

AND IF this port is IN the EnterpriseDomain

Action: Block SNMP traffic (5)

(We will define "edgeInterface" role and "EnterpriseDomain" later in this note.)

A possible drawback of (5) is that it is activated by the arrival of a packet event. Such events will be VERY common, meaning that the Policy Engine will be doing a lot of work when most of the time, no policy action is needed.

The second case could be addressed as follows:

Event: A new port is going to be enabled

Condition: IF this interface implements the "edgeInterface" role AND IF this port is IN the EnterpriseDomain

Action: InstallFilter("SNMP traffic filter", "block") (6)

4.2. Use Case 2: VPC

4.2.1. Generic

In practice, a public cloud operator can virtualize the cloud resources into multiple isolated virtualized private clouds and provide them to different tenants. Such a Virtualized Private Cloud is referred to as a VPC. In a typical VPC provided by, e.g., Alibaba or Amazon, through a control portal, tenants can establish and manage their VPC networks easily, for instance, deploying or removing virtualized network devices (e.g., virtualized routers and virtualized switches), adjusting the topologies of VPC networks, specifying packet forwarding policies, and deploying or removing virtual services (e.g., load balancers, firewalls, databases, DNS, etc.). The network functionalities that the tenant can access are virtualized and actually could be performed by the VMs located on the

servers connected through physical or overlay networks. Note that the servers may be located in different data centers which are geographically distributed.

The manipulation of the virtualized VPC network may also affect the configuration of physical networks. For instance, when a tenant cloud networks and specify the policies to steer the traffics through different VPNs in different conditions. Note that the VPCs that the tenant may be located in different geographic regions and the VPNs to those VPCs may need to be generated at run time. newly deploys two VMs in the VPC which are located in different DCs, the VPC control mechanism may have to generate a VPN between two DCs for the internal VPC communication. Therefore, the control mechanism for a VPC should be able to adjust the underlying network when a tenant changes the network or service deployment of the virtual VPC network.

In addition, a VPC, often provides other value added services (e.g., database Services, DNS) for VMs in certain VPCs. The VMs and the value added services could be located in different DCs, or even provided by different vendors. VPNs are configured for the VPCs to provide connection to the internal services in a tenant's own DC or organization. The access of such services should be controlled. For instance, the VMs in a VPC can access the database services only when the tenant has deployed a database within its VPC through the control portal.

In many cases, a tenant may need to specify how the VPCs are connected to its enterprise cloud networks. For instance, a tenant wants to deploy multiple VPNs to connect the VPC with its private cloud networks and specify the policies to steer the traffics through different VPNs in different conditions. Note that the VPCs that the tenant may be located in different geographic regions and the VPNs to those VPCs may need to be generated at run time.

In addition, a VPC, often provides other value added services (e.g., database Services, DNS) for VMs in certain VPCs. The VMs and the value added services could be located in different DCs, or even provided by different vendors. VPNs are configured for the VPCs to provide connection to the internal services in tenant's own DC or organization, and to create and manage VPNs to internal services. The access of VMs to data resources should be controlled. For instance, the VMs in a VPC can access a database service only when the tenant has deployed a database service into its VPC through the control portal.

4.2.2. Example 1

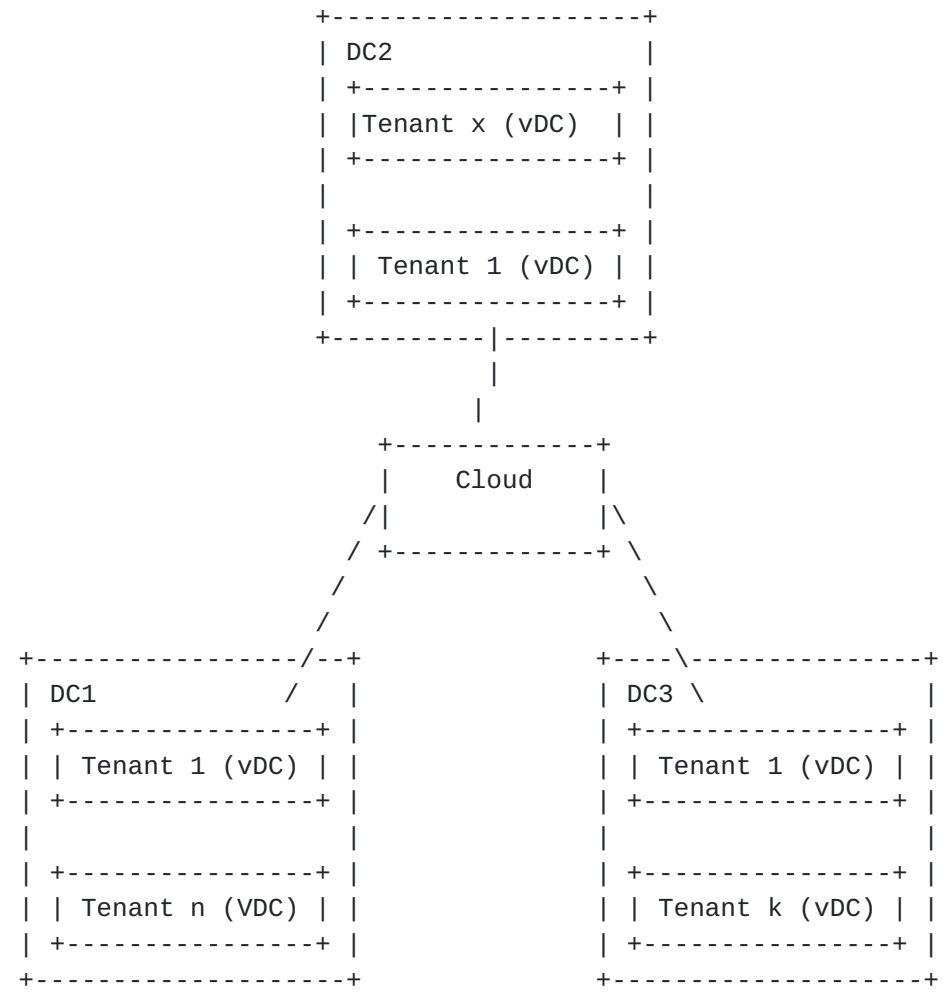

Figure 4: Resource Inter-connection for a VPC Tenant

When a cloud / DC operator signs a contract with customers, resource information such as network bandwidth, storage size, number of CPU, memory size, etc, will be specified.

But in deployment, the resources may be located in multiple distributed data centers, and tunnels will be created to connect these resources, which makes it look like one seamless entity - a virtual DC. There could be quite a number of tunnels, and the tunnels are dynamic, either for the reason of load balancing purpose or VM migration, or other reasons. This will make it difficult to configure the service statically or manually, service automation is very necessary.

The service management system will have a repository of available resources, including the topology. And also the management system

will have the customer specific information (location, SLA, agreed resources, etc).

The administrator can send the service requirement to the management system by a high level data model, which can further be mapped to low level detail data models, then finally mapped to configurations of network devices.

Target: Provide VPC service to customer A with specified resources and function (storage, computing, DNS, etc)

Declarative policy:

1. Allocate the required services on DCs according to a user's profile

2. Services located in multiple distributed DCs must be interconnected via VPNs

3. The VPNs associated to the services provided for a user must match the user's profile in terms of latency, speed and bandwidth

4.2.3. Example 2

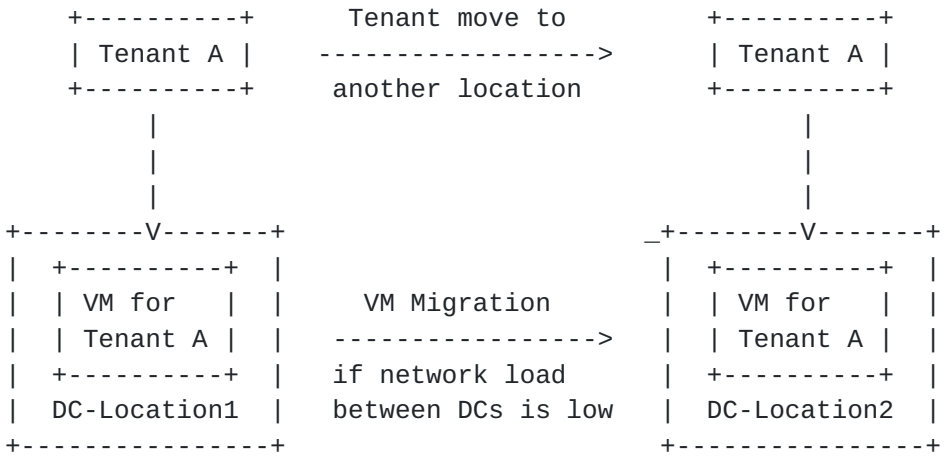

Figure 5: VM Migration if Tenant Move

As shown in the above figure, when a VPC tenant move from one location to another, where it is near to another DC, and the network load between the new DC and the previous DC is low, the tenant's VM should be migrated to the new DC in order for better user experience. After the VM is moved to the new DC, the network related to the VM must be updated accordingly.

Target: Perform VM migration when user location changed and the network load between the DCs is low.

ECA Policy:

Event: a VPC user's location is changed (near to another DC).

Condition: network_load(DC_old, DC_new) < threshold.

Action:

1. Migrate the VM to the new data center (DC_new).

2. Update the VPNs connecting the user's services.

In the above model it is assumed that the network management/ controller has the network topology, including attributes of the links, such as bandwidth. The network management/controller also monitors the real-time load on the links in the network topology.

The user's location can be identified by the user's IP address. When a user login, the network management/controller will check the user's IP address against an IP address database, such as the IP address assignments by IANA.

The network management/controller also maintain a mapping of DCs and IP address segments, say, a DC should serve users in a near location which can be identified by IP address segments. Though this is not always the case, sometimes the geographical distribution of network resource will also need to be considered besides the location (IP address). But, anyway, a mapping of DC and the IP address it should serve should be maintained.

If the controller detects a location change and a new DC is possible for the user, and the network load between the new DC and the old DC is low, then VM migration will be triggered and related network configuration will be performed.

4.3. Use Case 3: Instant VPN

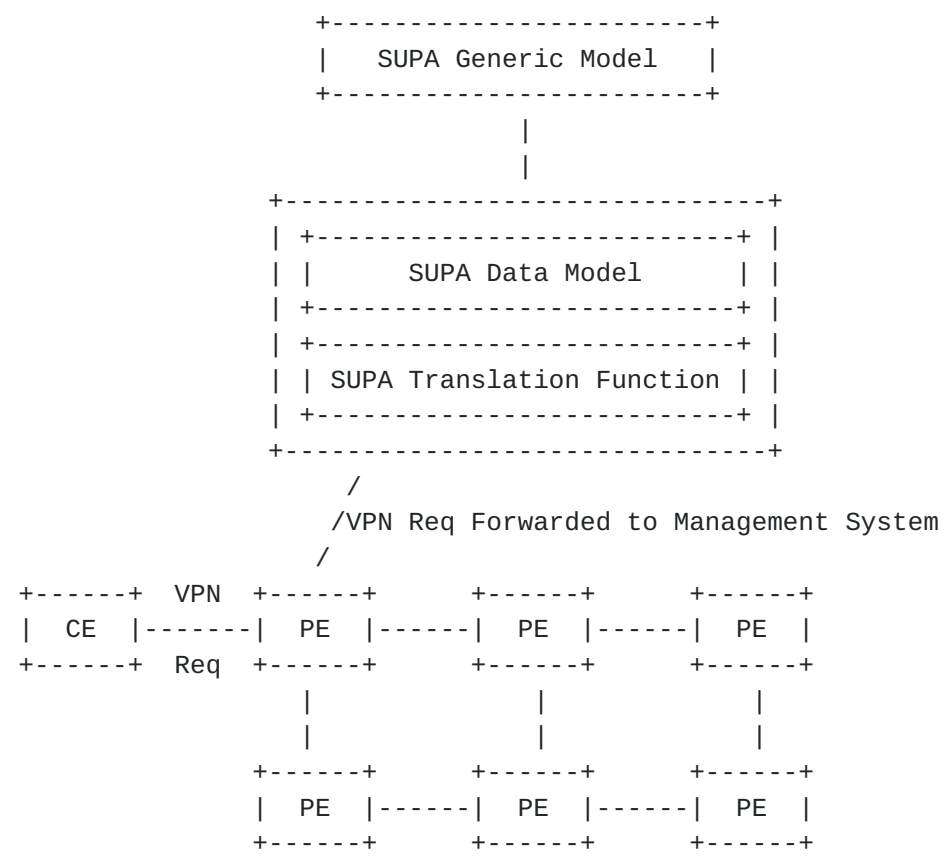

Figure 6: Instant VPN

Traditionally, when an operator needs to deploy VPN services for an enterprise customer, they will send a service staff to the customer site and make the wire connection between the CE and PE. The service staff will also collect the configuration information, e.g. port/frame/slot of PE, PE ID, etc, and then send the collected information back to the management system. The management system will configure the network according to this information as well as the customer' information (such as bandwidth, SLA, etc). The problem of this approach is that the service staff needs to collect the connection information and feedback to the management system, and MUST make sure the information matches the actual connection. This process is error prone.

New approach should not count on the physical / geographical information feedback by the service staff, minimize the operation procedures. The CE should send authentication (with credentials) request to the PE, and PE should forward the request to the management system together with port/frame/slot on which the request is received, the PE ID etc.

Target: Configure VPN for an enterprise customer to connect its enterprise network with VPC

ECA Policy:

Event: service management system receive a CE request for VPN creation (forwarded by PE).

Condition: Authentication and Authorization results are OK.

Action: Configure VPN based on received request, including the user's grade and physical info (port/slot/frame/route id, etc, from which the request is received).

5. IANA Considerations

This document makes no request of IANA.

Note to RFC Editor: this section may be removed on publication as an RFC.

6. Security Considerations

Since SUPA models can be used to generate configurations for network elements, the management applications which send models to service management system must go through authentication and authorization.

The handling of confliction of different polcies is out of scope of this memo.

7. Acknowledgements

This document has benefited from reviews, suggestions, comments and proposed text provided by the following members, listed in alphabetical order: Joel M. Halpern, Juergen Schoenwaelder, John Strassner, James Huang, Georgios Karagiannis.

8. References

8.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <http://www.rfc-editor.org/info/rfc6020>.

8.2. Informative References

[I-D.ietf-supa-generic-policy-data-model]
Halpern, J. and J. Strassner, "Generic Policy Data Model for Simplified Use of Policy Abstractions (SUPA)", draft-ietf-supa-generic-policy-data-model-02 (work in progress), October 2016.

[I-D.ietf-supa-generic-policy-info-model]
Strassner, J., Halpern, J., and S. Meer, "Generic Policy Information Model for Simplified Use of Policy Abstractions (SUPA)", draft-ietf-supa-generic-policy-info-model-02 (work in progress), January 2017.

[I-D.ietf-supa-policy-based-management-framework]
LIU, S., Strassner, J., Karagiannis, G., Klyus, M., Bi, J., and C. Xie, "SUPA policy-based management framework", draft-ietf-supa-policy-based-management-framework-00 (work in progress), August 2016.

Authors' Addresses

Ying Cheng (Editor)
China Unicom
No.21 Financial Street, XiCheng District
Beijing 100033
China

Phone: +86-010-66259394
Email: chengying10@chinaunicom.cn

Dapeng Liu
Alibaba Group
Beijing 100022
China

Email: max.ldp@alibaba-inc.com

Borui Fu (Editor)
China Telecom
Beijing
China

Email: fubr@ctbri.com.cn

Dacheng Zhang
Freelancer
Beijing
China

Email: Dacheng.zhang@gmail.com

Narasimha Vadrevu
VN Telecom Consultancy

Email: vadrevun@von20.com